ISSN: 2355-7699

JURNAL
TEKNOLOGI INFORMASI & ILMU KOMPUTER

Volume 3 | Nomor 4 | Desember 2016 | Halaman 226-299

FILKOM
FAKULTAS ILMU KOMPUTER

J T I I K

JURNAL TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

Volume 3, Nomor 4, Desember 2016 **ISSN 2355-7699**

JTIIK diterbitkan oleh Fakultas Ilmu Komputer (FILKOM), Universitas Brawijaya sejak April 2014. JTIIK memuat naskah hasil-hasil penelitian di bidang Teknologi Informasi dan Ilmu Komputer.

Alamat Redaksi dan Tata Usaha

Jurnal Teknologi Informasi dan Ilmu Komputer
Fakultas Ilmu Komputer (FILKOM), Universitas Brawijaya
Jl. Veteran No. 8 Malang, 65145
Telp./Fax (0341) 577911
Email: jtiik@ub.ac.id
Website: http://www.jtiik.ub.ac.id

Redaksi mengundang penulis untuk mengirimkan naskah yang belum pernah diterbitkan di media manapun. Pedoman penulisan naskah terdapat pada bagian belakang jurnal. Naskah yang masuk akan dievaluasi secara *blind-review* oleh Mitra Bestari dan Redaksi Pelaksana.

J T I I K

JURNAL TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

Volume 3, Nomor 4, Desember 2016 **ISSN 2355-7699**

Redaksi JTIIK mengucapkan terimakasih yang sebesar-besarnya kepada para reviewer yang telah bersedia untuk meluangkan waktunya dalam melakukan review pada naskah-naskah yang masuk di JTIIK FILKOM UB.

MITRA BESTARI

1. Arif Muntasa (Universitas Trunojoyo Madura)
2. Barlian Henryranu Prasetio (Universitas Brawijaya, Malang)
3. Budi Darma Setiawan (Universitas Brawijaya, Malang)
4. Eka Mistiko Rini (Politeknik Negeri Banyuwangi)
5. Erick Fernando (STIKOM Dinamika Bangsa Jambi)
6. Indri Sudanawati Rozas (UIN Sunan Ampel Surabaya)
7. Heliza Rahmania Hatta (Universitas Mulawarman, Samarinda)
8. Nurfiana (Institut Informatika dan Bisnis Darmajaya Bandar Lampung)
9. Rahimi Fitri (Politeknik Negeri Banjarmasin)
10. Ratih Ayuninghemi (Politeknik Negeri Jember)
11. Robbi Rahim (Institut Teknologi Medan)
12. Wayan Firdaus Mahmudy (Universitas Brawijaya, Malang)

J T I I K

JURNAL TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

Volume 3, Nomor 4, Desember 2016 **ISSN 2355-7699**

DAFTAR ISI

Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) p-ISSN: 2355-7699
Vol. 3, No. 4, Desember 2016, hlm. 226-237 e-ISSN: 2528-6579

IMPLEMENTASI ALGORITMA GENETIKA DALAM OPTIMASI MODEL AHP DAN TOPSIS UNTUK PENENTUAN KELAYAKAN PENGISIAN BIBIT AYAM BROILER DI KANDANG PETERNAK

Fita Lathifatul Mu'asyaroh[1], Wayan Firdaus Mahmudy[2]

[1,2]Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]fita.lathifa@gmail.com, [2]wayanfm@gmail.com

(Naskah masuk: 1 Oktober 2016, diterima untuk diterbitkan: 26 Desember 2016)

Abstrak

Peternakan ayam *broiler* merupakan salah satu jenis usaha yang paling potensial dikembangkan. Pola yang biasa digunakan dalam pengembangan ayam *broiler* adalah pola kemitraan inti plasma. Pada pola ini perusahaan mitra meninjau layak tidaknya kandang peternak untuk mendapatkan bibit ayam *broiler*. Dalam penelitian ini ada beberapa penilaian kriteria yang dilakukan yaitu: riwayat peternak, tinggi kandang, jarak antar kandang, kelembapan, kekuatan kandang dan keamanan. Agar dapat memperoleh penilaian kelayakan kandang yang optimal, penelitian ini menawarkan solusi menggunakan algoritma genetika sebagai algoritma untuk penentuan kandang peternak dalam pengisian bibit ayam *broiler*. Data yang digunakan dalam penelitian adalah 46 data kandang ayam *broiler*. Proses algoritma genetika ini menggunakan representasi *real–code* dengan panjang kromosom sesuai dengan kriteria yang ditentukan, metode crossover yang digunakan adalah *extended intermediate crossover*, metode mutasi yang digunakan adalah *random mutation*, dan diseleksi dengan metode *elitism*. Dari hasil pengujian yang diperoleh parameter optimal yaitu ukuran populasi 105 individu dengan rata-rata fitness sebesar 0,73910, generasi sebanyak 115 dengan rata-rata firness sebesar 0,7610 dan kombinasi cr 0,5 dan mr 0,1 dengan rata-rata fitness sebesar 0,75218. Hasil akhir berupa layak atau tidak layak kandang peternak untuk diisi ayam *broiler*.

Kata kunci: *Algoritma Genetika, Ayam Broiler, Optimasi.*

Abstract

Broiler breeders are one of the most business potential to be developed. The usual pattern has been used in the development of broiler chicken was a partnership plasma core. In this pattern, company partners reviewing the appropriateness of the cage breeders to obtain the seed of broiler chickens. In this study there was several assessment criteria, it was the history of breeders, high of cages, the distance between cages, moisture, strength and security of cages. In order to obtain optimal cage feasibility assessment, this studied offers a solution by used genetic algorithm as an algorithm for determining of cage breeders in seed filling of broiler chickens. The data used in this research is the 46 data of broiler chicken coop. The process of genetic algorithm using real-code representation the chromosome length in accordance with the prescribed criteria, the crossover method used is extended intermediate crossover, mutation method used is random mutation, and selected by the method of elitism. From the test results obtained optimal parameters such as the size of the population of 105 individuals with an average fitness of 0.73910, generation of 115 with an average of 0.7610 and combination of *cr* 0.5 and *mr* 0.1 with an average fitness of 0.75218. The final result is the properness for the breeder cage at the contents of broiler chickens.

Keyword : *Genetic Algorithm, Boiler Chickens, Optimization.*

1. PENDAHULUAN

Peternakan ayam *broiler* merupakan usaha yang memberikan konstribusi besar dalam penyediaan daging nasional untuk memenuhi kebutuhan protein hewani masyarakat(Bahari, et al., 2012). Ditinjau dari nilai gizinya, daging ayam *broiler* tidak kalah dibanding dengan daging dari ternak lain (Sholikin, 2011). Ayam *broiler* merupakan ayam penghasil daging yang memiliki beberapa keunggulan diantaranya, laju perputaran modal yang cepat dan waktu pemeliharaan yang singkat yaitu dalam lima minggu ayam broiler sudah dapat dipanen dengan bobot 1,5 kg/ekor(Maulana, 2008). Hal inilah yang mendorong banyak peternak mengusahakan peternakan ayam *broiler*. Saat ini peternak ayam *broiler* di kota Malang berkembang cukup pesat, hal tersebut dapat dilihat dari banyaknya kandang ayam *broiler* dibeberapa daerah yang menerapkan pola kemitraan inti plasma dengan perusahaan penyedia bibit ayam *broiler*.

Tujuan pola kemitraan ini adalah meningkatkan pendapatan, dan peningkatan skala usaha baik dari pihak perusahaan maupun peternak(Maulana, 2008). Sedangkan inti plasma yang dimaksud adalah

dimana kelompok peternak mitra bertindak sebagai plasma sedangkan perusahaan mitra sebagai inti. Ada beberapa persyaratan dalam mengikuti kemitraan seperti : peternak menyiapkan kandang, peralatan, mengajukan pendaftaran kerjasama dan wajib memberikan jaminan kepada perusahaan(Yunus, 2009). Kemudian dari pihak perusahaan akan meninjau kelayakan dari kandang yang akan diisi bibit ayam *broiler*(Imadudin, 2001). Fungsi kandang dalam berternak ayam *broiler* sangatlah penting karena kegagalan dalam beternak ayam pedaging tidak serta merta kesalahan anak buah kandang dalam memelihara ayam tetapi juga dapat disebabkan dari kelayakan kandang ayam tersebut. Memberikan bibit ayam secara sembarangan pada kandang yang dinilai kurang layak dapat menyebabkan kerugian dimasa mendatang baik bagi perusahaan maupun peternak, jika terdapat banyak ayam yang mati diakibatkan oleh ketidaknyamanan kandang maupun diakibatkan oleh faktor keamanan lingkungan tersebut.

Beberapa petugas penyuluhan lapangan selama ini masih menerapkan penilaian secara manual dan subjektif dalam menentukan kelayakan kandang untuk diisi bibit ayam *broiler*. Penilaian dilakukan perorangan dan penentuan kelayakannya secara subjektif, tidak mempertimbangkan keputusan dari pihak lain ataupun sistem. Perkembangan sistem informasi saat ini semakin banyak diminati, terutama dalam mendukung pengambilan keputusan.

Pada penelitian mengenai penentuan kelayakan pengisian kandang ayam broiler telah dilakukan sebelumnya oleh Indra dkk (2013), yang menjelaskan kegunaan metode Analytical Hierarchy Process (AHP) sebagai alat bantu dalam pengambilan keputusan pengisian bibit ayam *broiler*. Di dalam penelitian tersebut penilaian kelayakan kandang ayam *broiler* dapat ditentukan dari kriteria riwayat peternak, tinggi kandang, jarak antara kandang, kelembapan, kekuatan kandang dan keamanan. Pengujian akurasi dilakukan dengan mencocokkan hasil rekomendasi dari sistem dengan hasil rekomendari dari pihak lapang dalam penentuan layak atau tidaknya kandang untuk diisi bibit ayam *broiler*. Pengujian sensitivitas dilakukan untuk mengetahui kriteria yang berpengaruh dalam penentuan kelayakan kandang ayam *broiler* dengan melakukan penambahan dan pengurangan 10%, 20%, 30%, dan 40%. Hasil penelitian tersebut dengan menggunakan metode *Analitical Hirarchy Proses* (AHP) sebagai metode yang digunakan untuk menentukan bobot kriteria kandang dan Technique for Order Preference by Similarity to Ideal Solution(TOPSIS) yang digunakan untuk menentukan kelayakan kandang peternak, menghasilkan nilai akurasi sebesar 62,5% dan uji tingkat sensitivitas menunjukkan disetiap kriteria kandang memiliki selisih rata-rata sensitivitas yang hampir sama(Indra, 2013).

Pada penelitian ini digunakan algoritma evolusi untuk penentuan batas-batas tertentu. Terdapat banyak metode yang dapat digunakan dan termasuk dalam algoritma evolusi, diantaranya adalah *Simulated Annealing (SA)*, *Particle Swarm Optimization (PSO)*, *Evaluation Strategies (ES)*, *Genetic Algoritgm (GA)* dan lain sebagainya(Arnold, 2011). Dalam penentuan kelayakan pengisian bibit ayam *broiler* pada kandang peternak ini menggunakan algoritma genetika. Kelebihan metode Algoritma Genetika dibanding dengan metode algoritma evolusi lainnya adalah algoritma genetika dapat memecahkan suatu masalah yang kompleks (Mahmudy, Marian, Luong 2013). Selain itu algoritma genetika dapat memecahkan masalah optimasi dalam bidang *computer science* dengan tingkat kesuksesan yang tinggi(Restuputri, et al., 2014).

Dengan adanya sistem ini diharapkan dapat membantu dalam mengoptimalkan penilaian alternatif layak dan tidaknya beberapa kandang untuk diisi bibit ayam *broiler*. Sehingga dapat membantu peternak ayam *broiler* dalam meningkatkan produktivitas ternaknya dan meminimalisir kerugian yang bisa terjadi.

2. LANDASAN KEPUSTAKAAN

2.1 Kandang

Kandang adalah tempat tinggal ayam dalam melakukan semua aktivitasnya. Mulai dengan makan, minum dan tentu saja tumbuh maupun menghasilkan telur. Kandang yang tidak memenuhi persyaratan minimal tidak termasuk dalam arti kandang sebenarnya. Perlu sekiranya diperhatikan kenyamanan kandang sehingga mampu mendukung tercapainya performan ayam yang optimal(Rasyaf, 2003).

2.2 Sistem Kandang

Sistem kandang merupakan hal penting dalam pembuatan sebuah kandang ayam broiler. Syarat kandang ayam yang baik adalah kandang yang memenuhi standar yang telah ditentukan. Syarat-syarat kandang ayam harus dipenuhi adalah sebagai berikut (Rasyaf & Cahyono, 2004):

1. Kandang harus dibuat kuat agar dapat dipakai dalam waktu yang lama, dan tidak mudah roboh karena angin yang kencang.
2. Dapat menahan air hujan dan terik matahari langsung masuk kandang, tepi atap sebaiknya dibuat cukup lebar yaitu sekitar 1,25 meter dari dinding kandang.
3. Kandang tidak rapat tetapi harus terbuka, memiliki celah-celah yang terbuka yang dibuat dari anyaman bambu, kawat ram atau jeruji-jeruji bambu sehingga hewan pemangsa tidak dapat masuk melalui celah yang terbuka tersebut.
4. Ruang ventilasi dapat ditambahkan dengan membuat sistem atap monitor dan dapat

menggunakan kipas angin yang berfungsi menyedot udara kotor dalam kandang atau mengalirkan udara segar masuk ke dalam kandang.

5. Lantai kandang sebaiknya disemen agar memudahkan dalam pembersihan kandang dan dibuat lebih tinggi dari tanah disekitarnya.
6. Ukuran/luas kandang tergantung dari jumlah ayam yang akan dipelihara. Sebagai pedoman, kepadatan ayam dewasa per meter persegi adalah 10 ekor.
7. Selokan/parit sebaiknya dibuatkan disekeliling kandang. Hal ini penting agar pembuangan air tidak menggenang.
8. Tata letak kandang hendaknya dibangun diatas tanah yang lebih tinggi dari tanah sekitarnya agar udara dapat berputar dan bergerak bebas elintasi kandang sehingga peredaran uadara dapat berjalan dengan baik. Kandang tidak terletak pada lokasi yang sibuk dan gaduh mengingat ayam mudah stres, ukuran dan luas kandang disesuaikan dengan jumlah dan umur ayam.
9. Jarak antara kandang juga haus mendapat perhatian karena dapat mempengaruhi sirkulasi udara, tingkat kelembapan, dan tepreratur.
10. Tinggi kandang berkaitan erat dengan bedarnya kandang untuk kondisi indonesia. Ketinggian dari lantai sampai atap teratas minimal 6 meter, sedangkan ketinggian dari lantai sampai atap terendah minimal 3 meter. Ketinggian kandang mempengaruhi ventilasi, temperatur dan biaya.

2.3 *Analytical Hierarchy Process*(AHP)

Sub bab pada metode AHP yang dibahas meliputi konsep dasar AHP dan prosedur AHP. Metode ini digunakan sebagai model inputan.

2.2.1. Konsep Dasar AHP

AHP adalah sebuah hierarki fungsional dengan input utamanya persepsi manusia. Dengan hierarki, suatu masalah kompleks dan tidak terstruktur dipecahkan ke dalam kelompok-kelompok tersebut diatur menjadi dua bentuk hierarki. Model AHP memakai persepsi manusia yang dianggap "pakar" sebagai input utamanya. Pakar adalah seorang individu yang memiliki pengetahuan khusus pemahaman, pengalaman dan metode – metode yang digunakan untuk memecahkan persoalan dalam bidang tertentu(Nurcholis & Achlison, 2014).

Dalam menyelesaikan persoalan AHP ada beberapa prinsip dasar yang harus dipahami antara lain (Saaty & Vargas, 2006):

a. *Decomposition*, setelah mendefinisikan permasalahan atau persoalan, maka perlu dilakukan dekomposisi, yaitu memecah persoalan yang utuh menjadi unsur-unsur terkecil.
b. *Comparatif Judgement*, prinsip ini berarti membuat penilaian tentang kepentingan relatif dua elemen pada suatu tingkatan tertentu yang berkaitan dengan tingkatan diatasnya. Penilaian ini merupakan inti dari AHP, karena akan berpengaruh terhadap prioritas elemen-elemen lainnya. Hasil dari penelitian ini lebih mudah disajikan dalam bentuk matriks *Pairwise Comparison.* Bentuk matriks *Pairwise Comparison* dapat dilihat pada Tabel 1.

Tabel 1. Skala Penilaian Perbandingan Berpasangan

Intensitas Kepentingan	Keterangan
1	Kedua elemen sama pentingnya. Dua elemen mempunyai pengaruh sama besar.
3	Elemen yang satu sedikit lebih penting dari pada lemen yang lainnya. Pengalaman dan penilain sedikit mendukung satu elemen dibandingkan elemen lainnya.
5	Elemen yang satu lebih penting dari pada elemen yang lainnya. Pengalaman dan penilaian sangat kuat menyokong satu elemen dibandingkan elemen yang lainnya.
7	Satu elemen jelas lebih mutlak penting dari pada elemen lainnya. Satu elemen yang kuat disokong dan dominan terliat dalam prektek.
9	Satu elemen mutlak penting dari pada elemen lainnya. Bukti yang mendukung elemen yang satu terhadap elemen lain memiliki tingkat penegasan tertinggi yang mungkin menguatkan.
2, 4, 6, 8	Nilai – nilai antara dua pertimbangan yang berdekatan. Nilai ini diberikan bila ada dua kompromi diantara 2 pilihan.

Sumber : (Saaty & Vargas, 2006)

c. *Synthesis of Priority*, dari *matriks pairwise comparison vektor eigen* ciri-nya untuk mendapatkan prioritas lokal, karena *matriks pairwise comparison* terdapat pada tingkat lokal, maka untuk melakukan secara global harus dilakukan sintesis diantara prioritas lokal. Prosedur melakukan sintesis berbeda bentuk hierarki.
d. *Local Consistency*, konsistensi memiliki dua makna. Pertama, bahwa objek-objek yang serupa dapat dikelompokkan sesuai dengan keseragaman dan relevansinya. Kedua, tingkat hubungan antara objek-objek yang didasarkan pada kriteria tertentu.

2.2.2. Prosedur *Analytical Hierarchy Process*

Secara umum langkah-langkah yang harus dilakukan dalam menggunakan AHP untuk pemecahan suatu masalah adalah sebagai berikut (Saaty & Vargas, 2006) :

1. Mendefinisikan masalah dan menentukan solusi yang diinginkan, lalu menyusun hierarki dari permasalahan yang dihadapi.
2. Menentukan prioritas elemen
   a. Langkah pertama dalam menentukan prioritas elemen adalah membuat perbandingan pasangan, yaitu membandingkan elemen secara berpasangan sesuai kriteria yang diberikan.
   b. Matriks perbandingan berpasangan diisi menggunakan bilangan untuk merepresentasikan kepentingan relatif dari suatu elemen terhadap elemen yang lainnya.
3. Sintesis
   Pertimbangan – pertimbangan terhadap perbandingan berpasangan disintesis untuk memperoleh keseluruhan prioritas. Hal – hal yang dilakukan dalam langkah ini adalah :
   a. Menjumlahkan nilai-nilai dari setiap kolom pada matriks.
   b. Membagi setiap nilai dari kolom dengan total kolom yang bersangkutan untuk memperoleh normalisasi matriks.
   c. Menjumlahkan nilai-nilai dari setiap baris dan membaginya dengan jumlah elemen untuk mendapatkan nilai rata-rat.
4. Mengukur konsistensi
   Dalam pembuatan keputusan, penting untuk mengetahui seberapa baik konsistensi yang ada karena kita tidak menginginkan keputusan berdasarkan pertimbangan dengan konsistensi yang rendah. Hal – hal iyang dilakukan dalam langkah ini adalah sebagai berikut :
   a. Kalikan setiap nilai pada kolom pertama dengan prioritas relatif elemen pertama, nilai pada kolom kedua dengan prioritas relatif yang bersangkutan.
   b. Jumlahkan setiap baris.
   c. Hasil dari penjumlahan baris dibagi dengan elemen prioritas relatif yang bersangkutan.
   d. Jumlahkan hasil bagi diatas dengan banyaknya elemen yang ada, hasilnya disebut λ maks.
5. Hitung Konsistensi Index (CI), berikut persamaan konsistensinya :
   $CI = \frac{(\lambda \max - n)}{n-1}$ .....(2-1)
   Dimana n = banyaknya elemen.
6. Hitung Konsistensi Ratio (CR), berikut Persamaan Perhitungan Rasio konsistensi:
   $CR = \frac{CI}{RI}$ .....(2-2)
   Keterangan
   CR = *Consistency Ratio*
   CI = *Consistency Index*
   RI = *Indeks Random Consistency*
7. Memeriksa konsistensi hierarki. Jika nilainya lebih dari 10%, maka penilaian data judgement harus diperbaiki. Namun jika Ratio Konsistensi (CI/RI) kurang atau sama dengan 0,1 maka hasil perhitungan bisa dinyatakan benar (Saaty & Vargas, 2006)

2.3 Penentuan Kelayakan Kandang dengan *Technique for Order Preference by Similarity to Ideal Solution* (TOPSIS)

Sub bab ini membahas tentang penentuan layak tidak layaknya kandang. Metode yang digunakan adalah metode *Technique for Order Preference by Similarity to Ideal Solution* (TOPSIS). Bobot kriteria yang diperoleh dari metode AHP akan dijadikan acuan pada metode TOPSIS.

2.3.1 Konsep Dasar dengan *Technique for Order Preference by Similarity to Ideal Solution* (TOPSIS)

TOPSIS merupakan salah satu metode pengambilan keputusan multikriteria yang pertama kali dikenalkan oleh Yoon dan Hwang pada tahun 1981 (Juliyanti & I, 2011). Metode TOPSIS banyak digunakan dalam beberapa model Multiple Attribute Decision Making (MADM) dikarenakan metode ini memiliki beberapa keunggulan yaitu :

1. Konsepnya sederhana dan mudah dipahami.
2. Komputasinya efisien.
3. Memiliki kemampuan untuk mengukur kinerja relatif dari alternatif-alternatif keputusan dalam bentuk matematis yang sederhana.

2.3.2 Prosedur TOPSIS

Berikut langkah-langkah prosedur TOPSIS (Lestari, 2011):

1. Menentukan matrik keputusan yang ternormalisasi
   TOPSIS membutuhkan rating kriteria kelayakan setiap calon kandang ayam pada setiap kriteria atau subkriteria yang ternormalisasi. Berikut persamaan matriks ternormalisasi dapat dilihat pada Persamaan (2-3).
   $r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}^2}}$.....(2-3)
   r_{ij} = Normalisasi matrik
   x_{ij} = Nilai data pada baris ke i dan kolom ke j
   $\sqrt{\sum_{i=1}^{m} x_{ij}^2}$ = Akar dari jumlah baris ke i kolom ke j di kuadratkan

2. Menghitung matriks keputusan ternormalisasi terbobot
   Dalam menghitung matriks ternormalisasi terbobot, harus ditentukan terlebih dahulu nilai bobot yang merepresentasikan preferensi absolute dari pengambilan keputusan. Nilai

preferensi menunjukkan tingkat kepentingan relatif setiap kriteria atau subkriteria. Berikut perhitungan matrik ternormalisasi terbobot ditunjukkan pada Persamaan 2-4 dan Persamaan 2-5 digunakan untuk menghitung perkalian bobot preferensi degan matrik ternormalisasi atau matrik keputusan ternormalisasi terbobot. $w = w_1, w_2, w_3, \dots, w_n$.....(2-4)

$y_{ij} = w_i \times r_{ij}$.....(2-5)

w = bobot prioritas
y_{ij} = Matrik ternormalisasi terbobot
w_i = bobot prioritas ke i
r_{ij} = Matrik ternormalisasi

3. Menghitung matrik solusi ideal positif dan matriks solusi ideal negatif.
   Solusi ideal positif dan solusi ideal negatif dapat ditentukan berdasarkan rating bobot ternormalisasi. Persamaan solusi ideal positif dan solusi ideal negatif dapat dilihat pada Persamaan 2-6 dan Persamaan 2-7.

   $A^+ = (y_1^+, y_1^+, y_1^+ \dots y_n^+)$.....2-6)

   $A^- = (y_1^-, y_1^-, y_1^- \dots y_n^-)$.....(2-7)

   A^+= Solusi ideal positif/nilai maksimum dari matrik ternormalisasi terbobot
   A^-= Solusi ideal negatif/nilai minimum dari matrik ternormalisasi terbobot
4. Menghitung jarak antara nilai setiap alternatif dengan matrik sousi ideal positif dan matrik solusi ideal negatif.
   Jarak antara alternatif dan solusi ideal negatif terdapat pada Persamaan 2.9.
   Penghitungan jarak antara alternatif dengan solusi ideal positif terdapat pada Persamaan 2.8

   $D_i^+ = \sqrt{\sum_{j=1}^{n}(y_i^+ - y_{ij})^2}$.....(2-8)

   Keterangan :
   D_i^+ = Jarak antara laternatif dengan solusi ideal positif
   $\sqrt{\sum_{j=1}^{n}(y_i^+ - y_{ij})^2}$ = Akar dari jumlah nilai max dikurangi nilai matrik ternormalisasi terbobot
   Perhitungan jarak antara alternatif dan solusi ideal negatif terdapat pada Persamaan 2.9.

   $D_i^- = \sqrt{\sum_{j=1}^{n}(y_{ij} - y_j^+)^2}$.....(2-9)

   Keterangan :
   D_i^- = Jarak antara laternatif dengan solusi ideal positif
   $\sqrt{\sum_{j=1}^{n}(y_{ij} - y_j^+)^2}$ = Akar dari jumlah nilai matrik ternormalisasi terbobot dikurangi nilai max
5. Menghitung nilai preferensi untuk setiap alternatif
   Penghitungan nilai preferensi ditampilkan pada Persamaan 2.10.

   $V_1 = \frac{D_i^-}{D_i^- + D_i^+}$.....(2.10)

   Keterangan :
   V_1 = Nilai preferensi
   D_i^- = jarak antar solusi ideal negatif
   D_i^+ = jarak antar solusi ideal positif

Dari hasil perhitungan diatas nantinya dapat diketahui alternatif kandang yang layak maupun tidak layak untuk diberi bibit ayam broiler. Matode ini menggunakan inputan dari metode AHP sebagai bobot prioritas.

2.4 Algoritma Genetika

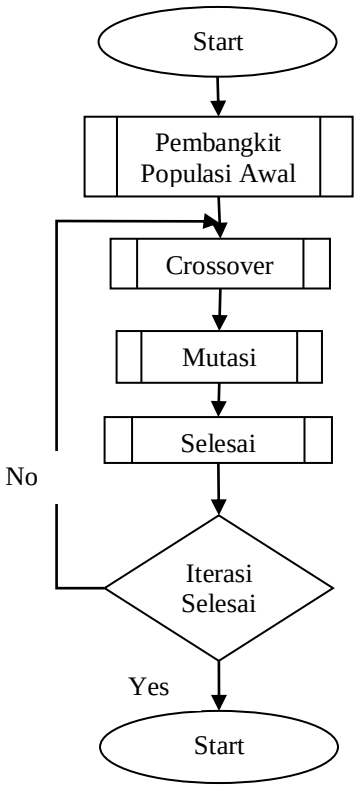
Algoritma genetika merupakan tipe *Evolution Algorithm* (EA) yang paling populer. Algoritma genetika berkembang seiring dengan perkembangan teknologi informasi yang sangat pesat. Karena kemampuannya untuk menyelesaikan berbagai masalah kompleks, algoritma ini banyak digunakan dalam bidang fisika, biologi, ekonomi, sosiologi dan lain-lain yang sering menghadapi masalah optimasi yang model matematikanya kompleks atau bahkan sulit dibangun (Mahmudy, 2013).

Dalam penyelesaian suatu masalah, algoritma genetika memetakan (*encoding*) suatu masalah menjadi string kromosom. String kromosom ini tersusun atas sejumlah *gen* yang menggambarkan variable-variable keputusan yang digunakan dalam solusi. Representasi string kromosom beserta fungsi *fitness* untuk menilai seberapa bagus sebuah kromosom untuk menjadi solusi yang layak sehingga dapat dimasukkan ke algoritma genetika (Mahmudy, 2013).

Proses dalam algoritma genetika dimulai dengan tahap inisialisasi, yaitu menciptakan individu – individu secara acak yang memiliki susunan gen (kromosom) tertentu yang mewakili solusi dari permasalahan. Tahap selanjutnya adalah reproduksi yang menghasilkan *offspring* dari individu yang ada dipopulasi. Setelah proses reproduksi dilakukan, lahir individu baru yang menyebabkan jumlah individu bertambah. Setiap kromosom mempunyai nilai *fitness*, dimana semakin besar nilai *fitnes* maka semakin baik kromosom tersebut untuk dijadikan calon solusi. Tahap terakhir adalah proses seleksi yaitu memilih individu dari himpunan populasi dan *offspring* yang dipertahankan hidup pada generasi berikutnya (Mahmudy, 2013).

Setelah melewati sekian iterasi, akan didapatkan individu terbaik. Individu terbaik ini akan mempunyai susunan kromosom yang bisa

dikonversi menjadi solusi yang paling baik atau mendekati optimum. Dapat disimpulkan bahwa algoritma genetika menghasilkan suatu solusi optimum dengan melakukan pencarian di antara sejumlah alternatif titik optimum berdasarkan fungsi *probabilistic* (Mahmudy, 2013). *Flowchart* cara kerja algoritma genetika dapat dilihat pada gambar 2.2.



Gambar 2 1 ***Flowchart Algoritma Genetika***
Sumber : Ariwibowo, 2008

2.5 Parameter Algoritma Genetika

Penentuan algoritma genetika merupakan pekerjaan yang tidak mudah. Beberapa parameter algoritma genetika adalah ukuran populasi (*popSize*), banyaknya generasi, *crossover rate* (cr), dan *mutation rate* (mr). Jika nilai parameter lagoritma genetika semakin besar, maka hal ini akan meningkatkan kemampuan eksplorasi algoritma genetika dalam pencarian solusi terbaik. Namun hal ini membuat waktu komputasi berlangsung lama karena kemungkinan algoritma genetika akan mengekporasi area yang tidak mempunyai nilai optimum (Mahmudy, 2013).

Tidak adanya metode pasti dalam penentuan nilai parameter algoritma genetika membuat nilai parameter sangat dipengaruhi oleh permasalahan yang akan diselesaikan. Dalam penelitian optimasi menggunakan algoritma genetika, serangkaian pengujian pendahuluan diperlukan untuk mendapatkan kombinasi nilai parameter yang sesuai(Mahmudy, 2013).

2.6 Penerapan Algoritma Genetika

Algoritma yang digunakan dalam penelitian ini adalah algoritma genetika dengan pengkodean real (*real-coded genetic algorithms*). Terdapat beberapa tahapan dalam penerapan algoritma genetika, yaitu melakukan representasi kromosom, inisialisai, reproduksi yang terdiri dari proses *crossover* dan mutasi, evaluasi, lalu yang terakhir adalah proses seleksi. Berikut ini merupakan penjelasan tahapan dalam algoritma genetika.

2.6.1 Representasi Kromosom

Representasi kromosom merupakan proses pengkodean dari penyelesaian asli suatu permasalahan. Solusi dari suatu permasalahan harus dipetakan (*encoding*) menjadi string kromosom. String kromosom tersusun atas sejumlah gen yang menggambarkan variable – variable keputusan yang digunakan dalam solusi(Mahmudy, 2013). Terdapat berbagai cara untuk menentukan representasi kromosom, yaitu sebagai berikut (Imbar & Jayanti, 2011):

a. Representasi Biner
   Representasi yang paling sederhana dan paling umum dimana setiap gen hanya bernilai 0 dan 1, Contoh 1000111, 1000101, 1000100 dan seterusnya.
b. Representasi Integer
   Representasi yang bernilai bilangan bulat. Contoh : 29, 18, 21, 9 dan seterusnya.
c. Representasi *Real Code*
   Pada penelitian ini, peneliti menggunakan representasi *real code* karena mewakili ukuran dari masing – masing bahan pakan yang akan dioptimasi. Representasi yang membutuhkan tingkat ketelitian amat tinggi representasi ini bernilai bilangan real. Contoh : 65.65 – 88.18, 21.89 dan seterusnya.
d. Representasi Permutasi
   Representasi yang digunakan untuk masalah *scheduling, travel salesmen problem,* atau yang tidak termasuk dari ketiga representasi.

2.6.2 Inisialisasi

Inisialisasi dilakukan untuk membangkitkan himpunan solusi baru secara acak/random yang terdiri dari sejumlah string kromosom dan ditempatkan pada penampungan yang disebut dengan populasi. Dalam tahap ini, ukuran populasi (*popSize*) harus ditentukan. Nilai ini menyatakan jumlah individu/kromosom yang ditampung dalam populasi. Panjang setiap string kromosom (*stringLen*) dihitung berdasarkan presisi variable dari solusi yang dicari (Mahmudy, 2013).

2.6.3 Reproduksi

Reproduksi bertujuan untuk menghasilkan keturunan dari individu – individu yang ada di populasi. Himpunan keturunan ini akan ditempatkan dalam penampungan *offspring.*Dua operator genetika yang digunakan dalam proses ini adalah *crossover* dan mutasi.

2.6.3.1. Crossover

Crossover dilakukan dengan cara memilih dua induk (parent) secara acak dari populasi. Metode *crossover* yang digunakan adalah *extended intermediate crossover*, yaitu metode yang menghasilkan *offspring* dari kombinasi nilai dua induk. Dalam rasio *offspring* yang dihasilkan proses *crossover* terhadap ukuran populasi sehingga akan dihasilkan offspring sebanyak cr x *popSize.*

Misalkan P1 dan P2 adalah dua kromosom yang telah diseleksi untuk melakukan *crossover*, maka bisa dibangkitkan *offspring* C1 dan C2 dengan rumus *offspring* hasil *crossover* sebagai berikut (Mahmudy, 2013) :

C1 = P1 + α (P2 – P1)
C2 = P2 + α (P1 – P2).....(2-1)

Keterangan :
C1, C2 = *Child* 1, *Child* 2
P1, P2 = *Parent* 1, *Parent* 2
α = Dipilih secara acak pada range yang ditentukan. Misal pada interval [-0,25 ; 1,25]

Misalkan yang terpilih sebagai induk adalah P4 dan P9 pada Tabel 2.7, α = [0.1104, 1.2336] maka akan dihasilkan dua *offspring* (C1 dan C2) sebagai berikut:

C1 : x1 = 5,8114 + 0,1104 (9,4374 – 5,8114) = 6,2118
x2 = 5,0779 + 1,2336 (6,6919 – 5,0779) = 7,0690
C2: x1 = 9,4374 + 0,1104 (5,8114 – 9,4374) = 9,0370
x2 = 6,6919 + 1,2336 (5,0779 – 6,6919) = 4,700

Jika ditentukan cr = 0,4 maka ada 0,4 x 10 = 4 *offspring* yang dihasilkan dari proses *crossover*. Setiap *crossover* akan menghasilkan dua anak, maka terdapat dua kali operasi *crossover* yang akan menghasilkan dua *offspring* berikutnya, yaitu C3 dan C4.

2.6.3.2. Mutasi

Mutasi biasanya digunakan sebagai operator untuk menjaga keragaman populasi. Mutasi dilakukan dengan memilih satu induk secara acak dari populasi. Dalam tahap ini nilai tingkat mutasi (*mutation rate / mr*) harus ditentukan untuk menyatakan rasio *offspring* yang dihasilkan dari proses mutasi terhadap ukuran populasi sehingga akan dihasilkan *offspring* sebanyak mr x *popSize* (Mahmudy, 2013).

Metode mutasi yang digunakan adalah *random mutation* yang dilakukan dengan menambah atau mengurangi nilai gen terpilih dengan bilangan *random* yang kecil. Misalkan domain variable x_j adalah [min_j , max_j] dan *offspring* yang dihasilkan adalah C=[x'_1...x'_n], maka nilai gen *offspring* bisa dibandingkan dengan rumus gen hasil mutasi sebagai berikut (Mahmudy, 2013) :

$x'_i = x'_i + r(max_i - min_j)$.....(2-2)

Keterangan :
x'_i = Induk terpilih
max_i = Nilai random terbesar
min_j = Range, misalkan [-0,1 , 0,1].

Misal yang terpilih sebagai induk adalah P2 pada Tabel 2.7, gen yang terpilih nomor 2 (x2) dan r = -0,0584. Maka akan dihasilkan offspring (C5) sebagai berikut:

C5: x1 = 8,491 (tetap)
x2 = 2,5754 – 0,0584 (7,3 – 0,0) = 2,1491

Anggap ditentukan mr = 0,2 maka ada 0,2 x 10 =2 *offspring* yang dihasilkan dari proses mutasi. *Offspring* dianggap C6. Keseluruhan *offspring* yang dihasilkan dari proses repreduksi (*crossover* dan mutasi) dapat dilihat pada Tabel 2.

Tabel 2. Hasil Offspring

Offspring	Chromosome		$f(x_1,x_2)$
	x1	x2	
C1	6,2118	7,0690	22,2048
C2	9,0370	4,7008	22,2313
C3	7,1636	0,0000	15,4774
C4	7,5479	7,3000	9,3531
C5	8,4917	2,1494	31,0389
C6	-1,1238	1,7097	12,0177

Sumber : Mahmudy, 2013

2.6.4 Evaluasi

Evaluasi digunakan untuk menghitung *fitness* pada tiap kromosom. Nilai *fitness* merupakan suatu ukuran kualitas dari tiap kromosom. Semakin besar nilai nantinya akan dijadikan calon solusi. Pada kasus dalam pencarian nilai maksimal, seperti pada persamaan 2-3(Mahmudy, 2013):

Fitness = f(x).....(2-3)

Selain pencarian nilai maksimum, *fitness* juga digunakan dalam pencarian nilai minimum. Pada kasus pencarian nilai minimum, nilai *fitness* bisa dihitung dengan salah satu dari dua rumus *fitness* pencarian nilai minimum seperti pada persamaan 2-6 (Mahmudy, 2013):

Fitness = C – f(x)
Fitness = 1/ (f(x)).....(2-4)

2.6.5 Seleksi

Seleksi merupakan tahapan terakhir yang dilakukan untuk memilih individu dari himpunan populasi dan *offspring* yang akan dipertahankan hidup pada generasi berikutnya. Semakin besar nilai *fitness* kromosom, maka semakin besar peluang kromosom tersebut terpilih. Hal ini dilakukan agar terbentuk generasi berikutnya yang lebih baik dari generasi sekarang (Mahmudy, 2013). Ada beberapa metode seleksi yang dapat digunakan, yaitu :

a. Seleksi *Elitism*

Metode seleksi yang digunakan dalam penelitian ini adalah metode seleksi *elitism*. Metode seleksi *elitism* bekerja dengan cara mengumpulkan semua individu dalam populasi (*parent*) dan *offspring* dalam satu penampungan.

Metode ini melakukan seleksi pada individu – individu dalam penampungan berdasarkan nilai *fitness* tertinggi. Individu terbaik dalam penampungan akan lolos untuk masuk dalam

generasi berikutnya. Metode seleksi *elitism* menjamin individu yang terbaik akan selalu lolos (Mahmudy, 2013). *Pseudocode* seleksi elitism dideskripsikan seperti pada Gambar 1.

PROCEDURE Elitism Selection
Input :
POP : himpunan individu pada populasi
POP_size : ukuran populasi
OS : himpunan individu anak (*offspring*) hasil reproduksi menggunakan *crossover* dan mutasi
Output :
POP : himpunan individu pada populasi setelah proses seleksi selesai
/* gabungan individu pada POP dan OS ke dalam TEMP */
TEMP <- Merge (POP, OS)
/* urutan individu berdasarkan *fitness* secara ascending */
OrderAscending (Temp)
/* copy pop_size individu terbaik ke POP */
POP <- CopyBest (Temp, pop_size)
END PROCEDURE

Gambar 1. Pseudocode Seleksi Elitism
Sumber : (Mahmudy, 2013)

Himpunan populasi dan *offspring* dicari nilai *fitness*nya masing-masing. Nilai *fitness*nya himpunan populasi dan *offspring* dapat dilihat pada Tabel 3.

Tabel 3. Kumpulan Individu

Individu	x1	x2	fitness
P1	1,4898	2,0944	19,8206
P2	8,4917	2,5754	34,7058
P3	1,4054	6,3035	20,6707
P4	5,8114	5,0779	14,5624
P5	-1,8461	1,7097	11,5858
P6	4,0206	4,4355	24,7106
P7	-0,1634	2,974	19,653
P8	5,2742	0,7183	22,1813
P9	9,4374	6.6919	12,4694
P10	-4.5575	0,1679	28,4324
C1	6,2118	7,0690	22,2048
C2	9,0370	4,7008	22,2313
C3	7,1636	0,0000	15,4774
C4	7,5479	7,3000	9,3531
C5	8,4917	2,1494	31,0389
C6	-1,1238	1,7097	12,0177

Sumber: Mahmudy, 2013

Metode seleksi *elitism* memilih nilai *fitness* yang terbesar berdasarkan jumlah *popSize*, sehingga kumpulan individu yang tertahan hidup pada generasi berikutnya dapat dilihat pada Tabel 4.

Tabel 4. Individu Hasil Seleksi Elitism

Asal pada P(t)	P(t+1)	*Fitness*
P1	P1	28,201
C3	P2	23,747
P3	P3	22,596
P4	P4	22,363
P9	P5	21,309
C6	P6	19,656
C1	P7	18,931
P5	P8	18,552
P8	P9	18,077
C5	P10	17,096

Sumber : Mahmudy, 2013

b. Roulette Whell

Metode ini menghitung nilai probabilitas seleksi (*prob*) pada tiap individu berdasarkan nilai *fitness*-nya. Nilai *prob* akan menghasilkan probabilitas kumulatif (*probCum*) yang digunakan untuk melakukan proses seleksi. Langkah- langkah membentuk *roulette wheel* berdasarkan probabilitas kumulatif (Mahmudy, 2013):

- Menghitung total *fitness* keseluruhan dari himpunan populasi *parent* ditambah *offspring*. Misal *fitness* (Pi) merupakan nilai *fitness* individu ke-i. Rumus :
  $total\ fitness = \sum_{i=1}^{popSize} fitness(P_i)$.....(2-5)
- Menghitung nilai probabilitas seleksi (prob) tiap individu. Rumus :
  $prob_k = \frac{fitness\,(P_i)}{total\ fitness}$, i = 1, 2,3.,popSize..(2-6)
- Menghitung nilai probabilitas kumulatif tiap individu. Rumus :
  $probCum_i = \sum_{j=1}^{i} prob_j$,
  i=1,2,3,....., popSize.....(2-9)

Selanjutnya ketika sudah membentuk *roulette wheel* maka individu akan dipilih secara acak berdasarkan nilai probabilitas kumulatif tersebut.

c. Binary Tournament Selection

Binary Tournament Selection merupakan metode seleksi dengan melakukan perbandingan individu yang memiliki nilai fitness terbaik dari individu – individu yang terpilih secara acak. Individu yang terpilih tersebut, akan menjadi individu pada generasi selanjutnya.

Misalkan individu – individu yang terpilih secara acak adalah P1 dan P2. P1 mempunyai nilai *fitness* 12,6342 sedangkan P2 memiliki nilai *fitness* 13,5345. Metode ini akan membandingkan nilai *fitness* antar P1 dan P2. P2 memiliki nilai *fitness* lebih baik dari P1 sehingga P2 terpilih menjadi individu pada generasi selanjutnya.

d. *Replacement selection*

Metode ini merupakan metode seleksi dimana *offspring* menggantikan *parent* jika nilai *fitness offspring* lebih besar dari nilai *fitness parent*. Metode ini memiliki aturan berdasarkan cara reproduksinya (Mahmudy, 2013):

- Pada Proses mutasi, *offspring* akan menggantikan induknya jika mempunyai nilai

fitness offspring lebih baik dari *fitness* induknya.

- Pada proses *crossover*, *offspring* dihasilkan dari dua induk. *Offspring* akan menggantikan induk yang lemah dengan nilai *fitness offspring* lebih baik dari nilai *fitness* induk yang terlemah. Induk yang terlemah merupakan induk yang mempunyai nilai *fitness* terburuk dari dua induk yang menghasilkan *offspring* tersebut.

2.7 Akurasi

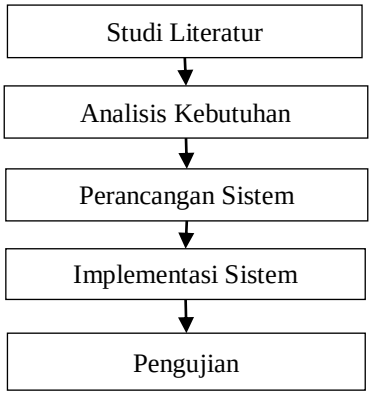
Akurasi adalah seberapa dekat suatu angka hasil pengukuran terhadap angka sebenarnya (*true value* atau *reference value*). Dalam penelitian ini akurasi diaknosa dihitung dari jumlah diaknosa yang tepat dibagi dengan jumlah data. Tingkat akurasi diperoleh dengan perhitungan sesuai dengan persamaan 2-11 (Adityo, 2013).

$$Tingkat\ akurasi = \frac{\sum data\ uji\ benar}{\sum total\ data\ uji} \text{.....(2-11)}$$

3. METODOLOGI PENELITIAN

3.1 Tahapan Penelitian

Dalam melakukan penelitian ini terdapat beberapa tahapan-tahapan yang dilakukan, hal tersebut akan dijelaskan pada Gambar 2.



Gambar 2. Tahapan - tahapan penelitian

3.2 Teknik Pengumpulan Data

Pengumpulan data dilakukan dengan penelitian tentang kelayakan kandang. Sistem ini digunakan untuk memberikan alternatif dalam membantu penentuan kelayakan kandang. Pengumpulan data pada penelitian ini menggunakan data primer. Data primer adalah data yang didapatkan langsung dari sumber penelitian. Pengumpulan data primer dapat dilakukan menggunakan wawancara, kuisioner maupun observasi.

3.3 Algoritma yang Digunakan

Penelitian ini menggunakan algoritma genetika, seperti yang sudah dijelaskan pada Bab 2 bahwa algoritma ini sudah dapat menyelesaikan beberapa permasalah seperti optimasi komposisi pakan sapi potong dan lain sebagainya(Kusuma, et al., 2015). Algoritma genetika merupakan tipe algoritma evolusi yang paling populer. Implementasi algoritma ini akan menggunakan bahasa Java karena program yang dikembangkan adalah program berbasis dekstop.

4. PERANCANGAN

Bagian ini menjelaskan tentang permasalahan yang akan diselesaikan dalam peneltian ini. Permasalahan dalam penelitian ini adalah bagaimana menentukan kelayakan kandang peternak untuk dapat diisi bibit ayam *broiler*. Dalam proses penentuan kelayakan kandang peternak terdapat 6 kriteria yang digunakan sebagai standar kelayakan kandang ayam *broiler*:

a. Riwayat peternak
   Status peternak menjadi sangat penting apakah peternak pernah mengalami kegagalan yang sering, selalu berhasil atau masih pemula.
b. Tinggi kandang
   Tinggi kandang merupakan kriteria pendukung dari kelayakan kandang tersebut, apakah sesuai dengan standart atau tidak. Semakin tinggi kandang tersebut semakin bagus.
c. Kekuatan kandang
   Kandang tersebut masih kokoh ataukah sudah hampir rusak atau bahan kandang tersebut sudah lapuk.
d. Sirkulasi udara(kelembapan)
   Sirkulasi udara ini mempengaruhi kelembapan, semakin lembab kandang tersebut semakin tidak bagus bagi bibit ayam *broiler*.
e. Jarak antar kandang
   Jarak antara kandang minimal terpisah selebar 1 kandang ayam, semakin terpisah jauh semakin bagus.
f. Keamanan
   Keamanan menjadi penting karena dalam peternakan ayam sering terdapat kejahatan dalam pencurian ayam *broiler*.

Misalkan sudah diketahui riwayat peternak berdasarkan range yang di tentukan (range peternak 1-5). Langkah selanjutnya adalah penilaian terhadap tinggi kandang, kekuatan kandang, sirkulasi udara pada kandang, jarak antar kandang, dan tingkat keamanan kandang.

Tabel 4. Contoh Penilaian Bobot Kriteria

ID	A	B	C	D	E	F
AC1	3.00	3.00	3.00	5.00	1.00	3.00
AC2	5.00	3.00	5.00	5.00	3.00	3.00
AC3	3.00	5.00	3.00	5.00	4.00	3.00
AC4	3.00	4.00	3.00	5.00	3.00	1.00

Keterangan :

A : Riwayat Peternak
B : Tinggi Kandang
C : Kekuatan Kandang
D : Kelembapan
E : Jarak Antar Kandang
F : Keamanan
ID : Identitas pemilik kandang

5. PENGUJIAN DAN PEMBAHASAN

Terdapat tiga pengujian yang dilakukan dalam pengujian ini yaitu pengujian yang akan dilihat dari hasil nilai *fitness* paling optimal, sehingga dapat ditemukan parameter terbaik yang digunakan untuk optimasi metode AHP dalam penentuan pengisian bibit kandang ayam *broiler*.

Proses pengujian dilakukan dengan melakukan tiga pengujian yaitu pengujian ukuran populasi, banyaknya generasi, dan kombinasi *cr* dan *mr*.

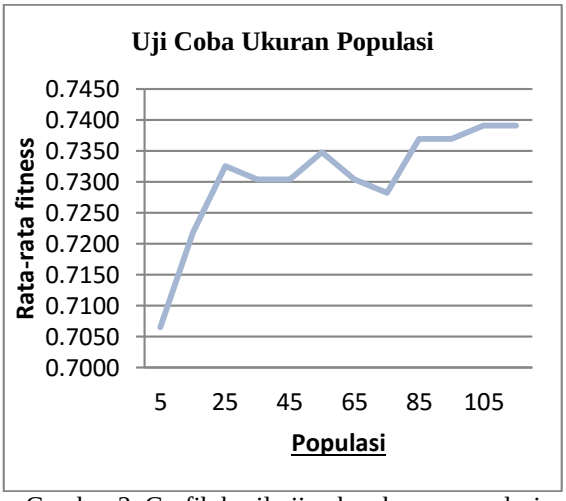
5.1. Pengujian dan Analisis Ukuran Populasi

Pengujian ukuran populasi digunakan untuk menentukan ukuran populasi yang terbaik untuk menghasilkan solusi terbaik dalam kasus ini. Berikut adalah parameter yang akan digunakan dalam pengujian.

a. Ukuran populasi : 5 – 115
b. Banyaknya generasi : 10
c. *Crossover Rate* : 0.5
d. *Mutation Rate* : 0.1

Pengujian dilakukan sebanyak sepuluh kali untuk mendapatkan hasil yang mewakili kemampuan algoritma secara utuh. *Fitness* terbaik pada setiap percobaan dihitung rata-ratanya untuk mengetahui ukuran populasi paling optimal. Hasil pengujian menunjukkan semakin besar ukuran populasi maka *fitness* yang dihasilkan cenderung semakin baik

Berdasarkan grafik hasil uji coba pada Gambar 3, ditunjukkan bahwa semakin besar ukuran populasi, maka rata – rata *fitness* yang dihasilkan cenderung meningkat. Pada grafik tersebut, dapat dilihat rata – rata *fitness* dari ukuran populasi 5 menuju ukuran 85 mengalami peningkatan, selanjutnya pada ukuran populasi diatas 105 cenderung stabil. Hal ini menunjukkan bahwa ukuran populasi 105 adalah ukuran populasi yang optimal. Perubahan yang tidak begitu besar ini terjadi karena anak yang dihasilkan pada proses reprodukasi mirip dengan induknya(Mahmudy, 2013).



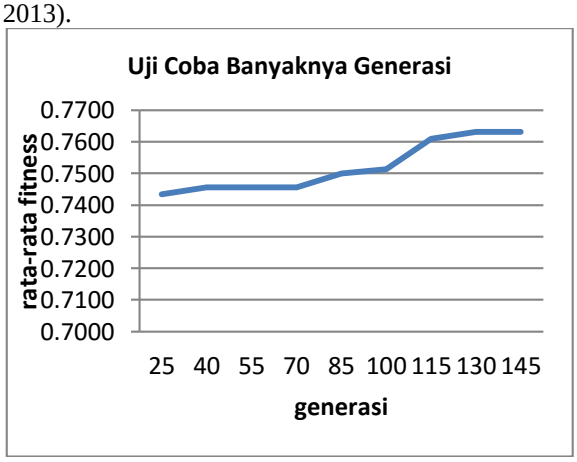
Gambar 3. Grafik hasil uji coba ukuran populasi

5.2. Pengujian dan Analisis Banyaknya Generasi

Pengujian banyaknya generasi dilakukan untuk menentukan banyaknya generasi yang dapat mengasilkan solusi terbaik dalam kasus ini. Pada pengujian banyaknya generasi ini, digunakan ukuran populasi 105 yang dianggap dapat menghasilkan nilai *fitness* paling optimal. Untuk lebih detailnya mengenai parameter yang digunakan pada uji coba banyaknya generasi adalah sebagai berikut:

a. Ukuran populasi = 105
b. Banyaknya generasi = 25 – 145
c. *Crossover Rate* = 0.5
d. Mutation Rate = 0.1

Berdasarkan grafik hasil uji coba pada Gambar 4, dapat dilihat rata-rata *fitness* mengalami peningkatan dari generasi 40 menuju generasi 100. Rata-rata *fitness* yang dihasilkan generasi diatas 115 cenderung stabil karena perubahan rata-rata *fitness* yang tidak begitu besar. Hal ini menunjukkan bahwa generasi sebanyak 100 adalah generasi yang paling optimal. Semakin banyak generasi maka semakin besar waktu komputasi, namun belum tentu menghasilkan solusi yang lebih baik (Mahmudy, 2013).



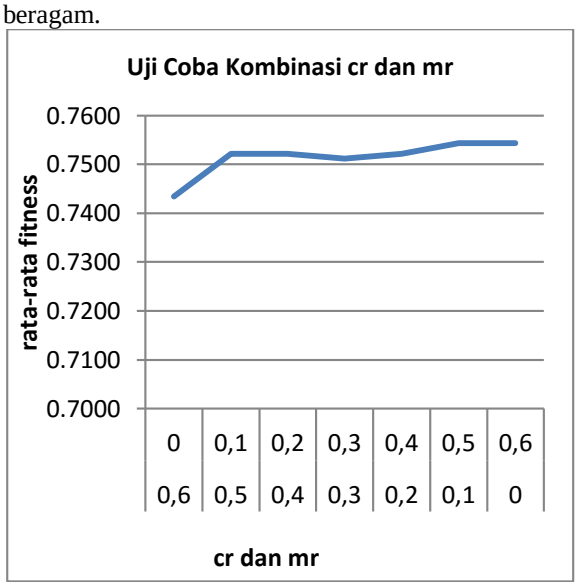
Gambar 4. Grafik hasil uji coba banyaknya generasi

5.3. Pengujian dan Analisis Kombinasi Cr dan Mr

Pengujian *crossover rate* (*cr*) dan *mutation rate* (mr) dilakukan untuk mengetahui kombinasi cr dan mr terbaik yang dapat menghasilkan fitness paling optimal. Nilai *cr* dan *mr* yabf digunakan antara 0 sampai 0.6 dan keduanya jika dijumlahkan menjadi 0.6. Pengujian cr dan mr juga menggunakan hasil pengujian sebelumnya yaitu hasil pengujian jumlah populasi dan generasi yang menghasilkan nilai *fitness* paling optimal. Untuk lebih detailnya mengenai parameter yang digunakan dalam uji coba kombinasi *cr* dan *mr* adalah sebagai berikut :

a. Ukuran populasi : 105
b. Banyaknya generasi : 115

Berdasarkan Grafik hasil pengujian pada Gambar 5, rata-rata *fitness* yang dihasilkan sangat beragam.



Gambar 5. Grafik hasil uji coba kombinasi cr dan mr

Permasalahan yang ingin diselesaikan dipengaruhi oleh kombinasi nilai parameter yang tepat(Mahmudy, 2013). Kombinasi *cr* dan *mr* yang dihasilkan pada setiap kasus akan menunjukkan hasil yang berbeda tergantung permasalahan yang akan diselesaikan. Hal ini disebabkan tidak adanya suatu ketepatan nilai kombinasi *cr* dan *mr* yang dapat digunakan untuk menghasilkan soluasi optimal.

5.4. Solusi Terbaik yang Pernah Didapat

Dalam penelitian optimasi menggunakan algoritma genetika, serangkaian pengujian pendahuuan diperlukan untuk mendapatkan kombinasi nilai parameter yang sesuai(Mahmudy, 2013). Pada penelitian ini , didapatkan beberapa parameter terbaik dengan rata-rata fitness paling optimal, yaitu ukuan populasi = 105, banyaknya generasi = 115, *crossover rate* = 0.5, dan *mutation rate* = 0.1. Nilai fitness yang dihasilkan dari parameter tersebut adalah 0,75218.

Setelah melakukan serangkaian uji coba, algoritma genetika dianggap mampu untuk menyelesaikan optimasi metode AHP dalam menentukan kelayakan kandang untuk pemberian bibit ayam broiler. Peternak dan petugas lapang peternakan ayam broiler dapat menggunakan rekomendasi ini untuk mengetahui tingkat kelayakan kandang ayam, sehingga dapat mengurangi kerugian dalam beternak ayam *broiler*.

6. PENUTUP

Berdasarkan hasil uji coba parameter algoritma genetika pada permasalahan optimasi metode AHP dalam penentuan kelayakan kandang ayam *broiler*, terdapat beberapa kesimpulan :

1. Metode AHP dapat diterapkan dalam penentuan kelayakan bibit ayam *broiler* pada kandang peternak dengan akurasi yang didapat yaitu 63.0 %.
2. Algoritma genetika dapat menyelesaikan permasalahan optimasi metode AHP dalam menentukan kelayakan kandang ayam *broiler*. Disini dapat dilihat dari sistem yang dapat menghasilkan nilai akurasi lebih tinggi dari penelitian sebelumnya.
   Parameter terbaik dengan rata-rata nilai *fitness* paling optimal yang didapatkan dari hasil pengujian adalah sebagai berikut :
   - Jumlah populasi : 105
   - Banyaknya generasi : 115
   - *Crossover Rate* : 0.5
   - *Mutation Rate* : 0.1
3. Pengukuran solusi dari permasalahan optimasi metode AHP ini dilakukan dengan perhitungan nilai fitness yang diperoleh dari hasil akurasi metode AHP dengan menggunakan bobot yang didapat dari inisialisasi kromosom.

Penelitian ini dapat dikembangkan untuk menyelesaikan masalah optimasil metode AHP, dengan menggunakan mengembangkan metode AHP seperti fuzzy AHP dan lain sebagainya. Selain itu penggunaan metode *crossover*, mutasi dan seleksi yang berbeda juga dapat mempengruhi hasil nilai *fitness* yang berbeda pada setiap individu.

7. DAFTAR PUSTAKA

Adityo, P., 2013. *Implementasi Metode AHP untuk Aplikasi Rekomendasi Peringkat Kinerja Guru Pada SMA NEGERI I MAOSPATI.* Malang: Tugas Akhir Ilmu Komputer Universitas Brawijaya.

Arnold, A., 2011. *Penerapan Algoritma Genetika Pada Penentuan Komposisi Pakan Ayam Petelur.* Indonesia: Universitas Pelita Harapan.

Bahari, Mustadjab, M. M., Hanani, N. & Nugroho, B. A., 2012. Analisis contract Farming Usaha Ayam Broiler. p. 109.

Imadudin, 2001. *Analisis Kemitraan Pola Perusahaan Inti-Rakyat (PIR) Usaa peternak Ayam Ras Pedaging*. Bogor: Jurusan Sosial Ekonomi Industri Peternakan Institite Pertanian Bogor.

Imbar, V. & Jayanti, 2011. *Implementasi Algoritma Genetika Pada Aplikasi Penjadwalan Dengan Studi Kasus Pada SMP X*. Bandung: Seminar Teknik Informatika dan Sistem Informasi, Fakultas Teknologi Informasi, Universitas Kristen Maranatha.

Indra, B. Y., 2013. *Sistem Pendukung Keputusan Penentuan Kelayakan Pengisian Bibit Ayam Broiler Dikandang Peternak Menggunakan Metode AHP dan TOPSIS*. Malang: Program Teknologi Informasi dan Ilmu Komputer. Universitas Brawijaya.

Juliyanti, I. M. & I, M., 2011. *Pemilihan Guru Berprestasi Menggunakan Metode AHP dan TOPSIS*. Yogyakarta, Prosding Seminar NAsional Penelitian, Pendidikan dan Penerapan MIPA Universitas Negeri Yogyakarta.

Kusuma, J. I., Mahmudy, W. F. & Indriati, 2015. Optimasi Komposisi Pakan Sapi Potong Menggunakan Algoritma Genetika. *Jurnal Mahasiswa PTIIK Universitas Brawijaya*, p. 7.

Lestari, S., 2011. *Seleksi Penerimaan Calon Karyawan Menggunakan Metode TOPSIS*. Bali: Konferensi Nasional Sistem dan Informatika .

Mahmudy, W. F., 2013. *ALgoritma Evolusi*. Malang: Program Teknologi Informasi dan Ilmu Komputer. Universitas Brawijaya.

Mahmudy, WF, Marian, RM & Luong, LHS 2013, 'Modeling and optimization of part type selection and loading problems in flexible manufacturing system using real coded genetic algorithms', *International Journal of Electrical, Computer, Electronics and Communication Engineering*, vol. 7, no. 4, pp. 251-260.

Maulana, M. L., 2008. Analisis Pendapatan Peternak Ayam Ras Pedaging Pola Kemitraan Inti Plasma.

Nurcholis & Achlison, U., 2014. *Sistem Pakar Diagnosa Penyakit dan Hama Tanaman Dengan Metode Forward Chaining Berbasis Multiuser*. Semarang: Sekolah Tinggi Elektronik dan Komputer (STEKOM).

Rasyaf & Cahyono, 2004. *Pengelolahan Peternakan Unggas Pedaging*. Jakarta: Kanisius.

Rasyaf, M., 2003. *Peternak Ayam Petelur*. Depok: PT. Penebar Swadaya.

Restuputri, B. A., Mahmudy, W. F. & Cholissodin, I., 2014. Optimasi Fungsi Keanggotaan Fuzzy Tsukamoto Dua Tahap Menggunakan Algoritma Genetika Pada Pemilihan Calon Penerima Beasiswa dan BBP-PPA (Studi Kasus: PTIIK Universitas Brawijaya Malang). *Jurnal Mahasiswa PTIIK Universitas Brawijaya*, Volume 5, p. 2.

Saaty, T. & Vargas, L., 2006. *Decision Making With The Analytic Network Process*. United State of America: springer.

Sholikin, H., 2011. Manajemen Pemeliharaan Ayam Broiler di Peternakan UD Hadi PS Kecamatan Nguter Kabupaten Sukoharjo. p. 01.

Yunus, R., 2009. *Analisis Efisiensi Usaha Pternakan Ayam Ras Pedaging Pola Kemitraan dan Mandiri di Kota Palu Provinsi Sulawesi Tengah*. Semarang: Program PAscasarjana Universitas Diponegoro.

Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) p-ISSN: 2355-7699
Vol. 3, No. 4, Desember 2016, hlm. 238-247 e-ISSN: 2528-6579

RANCANG BANGUN APLIKASI *GAME* EDUKASI PEMBELAJARAN AKSARA LAMPUNG "AJO DAN ATU - BELAJAR AKSARA LAMPUNG", BERBASIS ANDROID DENGAN SISTEM *MULTI-ENDING* MENGGUNAKAN *ENGINE* REN'PY

Gigih Forda Nama[1], Flesi Arnoldi[2]
[1]Teknik Informatika
[2]Teknik Elektro
Universitas Lampung
Email: [1]gigih@eng.unila.ac.id, [2]flesiarnoldi@gmail.com

(Naskah masuk: 1 Oktober 2016, diterima untuk diterbitkan: 26 Desember 2016)

Abstrak

Perkembangan teknologi informasi saat ini berkembang sangat pesat. Teknologi tersebut banyak dimanfaatkan untuk berbagai keperluan, antara lain agar dapat terhubung dengan kerabat yang jauh, mencari informasi, dan *game*. Pada penelitian ini dilakukan pembangunan *game* edukasi pembelajaran aksara Lampung menggunakan metode *extreme programming*, dengan alat bantu *Ren'Py engine. Game* yang dikembangkan berdasarkan kebutuhan calon pengguna untuk belajar aksara Lampung. Dari hasil iterasi menggunakan *extreme programming*, telah menghasilkan sebuah *game* edukasi yang memiliki sistem *multi-ending* dan sistem *point*. Sistem ini dapat membantu pengguna untuk belajar aksara Lampung, *game* edukasi ini dikembangkan untuk perangkat *smartphone* Android yang memiliki sistem operasi minimal versi 2.3+ (*gingerbread*), sehingga *game* ini dapat dimainkan dimanapun dan kapan pun.

Kata kunci: *game edukasi belajar aksara Lampung, multi-ending game, metode extreme programming*

Abstract

Nowadays the development of information technology rapidly increase. The technology widely used for various purposes, such as to communicate with other family, seeking information, and games. In this research, the development of Lampung alphabet educational game using extreme programming methods, with Ren'Py engine tools. The games was developed based on the needs of potential users that learn Lampung alphabet. From the results of the extreme programming iteration, has produced an educational game that has a multi-ending system and point system. This system can help the user to learn Lampung alphabet, this educational game developed for Android smartphone devices that have a minimal operating system version 2.3+ (Gingerbread), this game can be played anywhere and anytime.

Key words: *lampung alphabet educational game, multi-ending game, extreme programming method*

1. PENDAHULUAN

Pemerintah Provinsi Lampung mewajibkan seluruh sekolah dasar dan menengah untuk memberikan mata pelajaran bahasa Lampung kepada seluruh siswanya. Biasanya pada masing-masing sekolah sudah memiliki Guru bahasa Lampung, mereka berkewajiban dalam memberikan pemahaman terhadap aksara Lampung dan penggunaannya dalam kehidupan sehari-hari. Dalam kenyataannya para Guru terutama di tingkat dasar sering mendapati minat belajar siswa yang rendah terhadap pelajaran bahasa Lampung.

Berdasarkan hasil pencarian tim peneliti pada beberapa artikel baik melalui *internet* mapun perpustakaan daerah belum banyak ditemukan hasil-hasil penelitian metode terbaik dalam hal pengajaran bahasa Lampung. Padahal pada tingkat pendidikan dasar sangat mengandalkan penggunaan metode-metode yang aplikatif dan menarik. Pembelajaran yang menarik akan memikat anak-anak untuk terus dan betah mempelajari Bahasa Lampung. Apabila siswa sudah tertarik dengan pembelajaran maka akan membantu meningkatkan prestasi siswa dalam bidang bahasa. Di sebagian besar siswa, pembelajaran Bahasa Lampung dirasa membosankan karena mereka sudah merasa sulit menerima penyampaian materi karena dikemas kurang menarik sehingga secara tidak langsung siswa menjadi lemah dalam penangkapan materi tersebut, dan banyak anak anak sekarang yang

kurang memahami aksara Lampung. Selain faktor penyampaian materi, proses pembelajaran bahasa Lampung juga terkendala dengan sedikitnya media ajar yang tersedia di sekolah, yang sebagian besar hanya mengandalkan buku saja.

Berawal dari permasalahan tersebut, maka dirancanglah suatu permainan yang dapat mendukung siswa tingkat dasar dalam belajar aksara Lampung sehingga menjadi lebih menarik dan anak akan menjadi lebih tertarik untuk mempelajarinya. Kelebihan tersebut juga akan berpengaruh pada kualitas belajar anak sehingga anak tidak merasa bosan. Sistem *multi-ending* juga diterapkan ke dalam permainan, sehingga anak dapat memilih jalur cerita mereka sendiri dan akan menghasilkan akhir cerita yang berbeda setiap jalur cerita yang mereka pilih sendiri.

Permainan yang dikembangkan ini memiliki kelebihan berupa tampilan yang menarik, karakter – karakter dalam permainan dibuat lebih khas, memiliki sistem *multi-ending*, memiliki sistem *points*, dan adanya interaksi antara permainan dan pengguna. Selain itu, permainan ini dikemas ke dalam bentuk aplikasi telepon pintar atau *smartphone* berbasis Android, sehingga anak dapat memainkannya dimanapun dan kapan pun. Melalui aplikasi game edukasi ini diharapkan dapat menghasilkan terobosan baru dalam sistem belajar bahasa lampung yang lebih menarik dan efektif.

2. TINJAUAN PUSTAKA

2.1. Game Edukasi

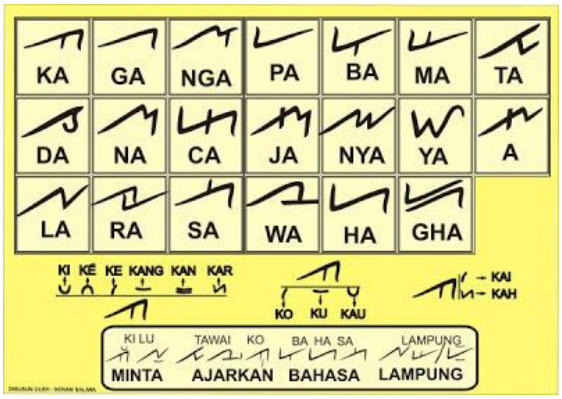
Game edukasi atau biasa juga disebut sebagai *edutainment*, adalah bagian dari permainan komputer pendidikan yang mudah dikenali dengan struktur *reward* atau penghargaan yang jelas dalam permainan terpisah dari pengalaman pendidikan (Egenfeldt, 2015). Sedangkan tujuan dari *game* edukasi adalah untuk melibatkan dan memotivasi pemain melalui pengalaman langsung dengan dunia *game*. Permainan harus memberikan kemungkinan reflektif menjelajahi fenomena, pengujian hipotesis dan membangun obyek, untuk membuat *game* yang baik, alur permainan sangat penting. Namun, selain itu diperlukan juga penciptaan alur cerita yang baik, grafis dan suara yang sesuai, dan keseimbangan dalam permainan. Jika aspek ini diabaikan, tentu saja hanya tidak akan cukup jika hanya mengandalkan alur permainan yang baik (Kiili, 2004).

2.2. Multi-Ending

Berdasarkan kamus definisi Oxford, *multi* atau *multiple* memiliki arti: mempunyai atau melibatkan beberapa bagian, elemen, atau anggota. Sedangkan *ending* memiliki arti: akhir atau bagian final dari sesuatu. Dari definisi – definisi tersebut, maka dapat disimpulkan bahwa *multi–ending* memiliki arti: memiliki beberapa bagian akhir. Jika sebuah *game* memiliki *multi–ending*, maka *game* tersebut mempunyai beberapa jenis cerita akhir atau cerita akhir yang bercabang.

2.3. Aksara Lampung

Had Lampung dipengaruhi dua unsur, yaitu Aksara Pallawa dan Huruf Arab. Had Lampung memiliki bentuk kekerabatan dengan aksara Rencong, Aksara Rejang Bengkulu, aksara Sunda, dan aksara Lontara. Had Lampung terdiri dari huruf induk, anak huruf, anak huruf ganda dan gugus konsonan, juga terdapat lambang, angka dan tanda baca. Had Lampung disebut dengan istilah Kaganga ditulis dan dibaca dari kiri ke kanan dengan Huruf Induk berjumlah 20 buah (Aryantio, 2015).



Gambar 1. Aksara Lampung (Aryantio, 2015)

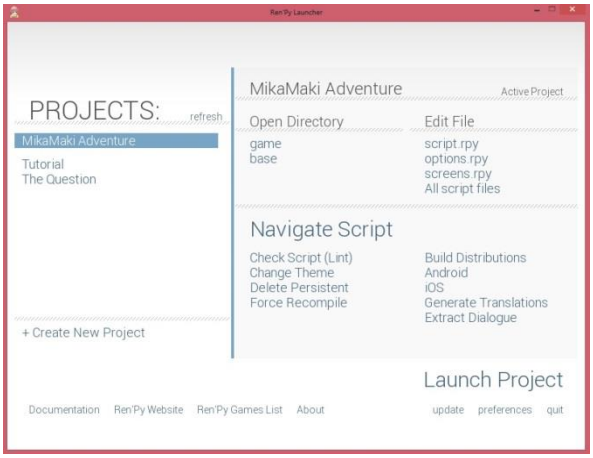
2.4. Python Programming Language

Python merupakan bahasa pemrograman yang saat ini tengah populer dan banyak digunakan oleh peneliti dalam membangun aplikasi, dalam penelitian yang dilakukan oleh Gigih Forda Nama (Nama, G.F. dkk, 2014-2015) Python digunakan sebagai *engine* utama yang digunakan dalam pemrosesan data, Python terbukti berjalan baik dan stabil dalam pengembangan aplikasi *web* maupun *mobile*.

2.5. Ren'Py

Ren'Py adalah sebuah *visual novel engine* yang digunakan oleh banyak orang dari seluruh dunia, yang membantu kita menggunakan kata, gambar dan suara untuk menceritakan sebuah cerita interaktif yang dapat dijalankan pada perangkat komputer dan perangkat *mobile* seperti *smartphone*. Cerita interaktif bisa berupa *visual novel* dan *game* simulasi. Bahasa pemrograman yang mudah, membantu pengguna untuk menulis *visual novel* yang yang besar secara efisien, dimana bahasa pemrograman Python sendiri sudah cukup untuk *game* simulasi yang kompleks (Ren'Py, 2015). Ren'py merupakan program *open source* dan gratis digunakan untuk kepentingan komersial sehingga pengguna tidak perlu membayar kepada pihak pengembang aplikasi untuk menjual *game* yang telah dibuat dengan menggunakan Ren'py. Salah satu kelebihan besar Ren'py yaitu *game* yang dibuat

dengan program ini dapat berjalan di hampir semua komputer. Tiga sistem operasi yang didukung oleh Ren'py yaitu Windows, Mac OS X, dan Linux, *game* yang dibuat dengan menggunakan Ren'py tidak tergantung *software* lain pada ketiga sistem operasi tersebut. Maka dari itu pengguna tidak perlu mengunduh *runtimes*, *drivers*, *codecs*, atau sejenisnya. Jika pemain mempunyai salah satu sistem operasi yang didukung, maka *game* yang dibuat oleh Ren'py dapat berjalan. Bahkan, jika tidak sekalipun, Ren'py ditulis dengan menggunakan teknologi *portable* seperti *pygame*, maka kemungkinan dapat dibuat untuk dijalankan (Ren'Py, 2015.



Gambar 2. *Main Interface Ren'Py*

2.6. IDE (Integrated Development Environment) pada Ren'Py

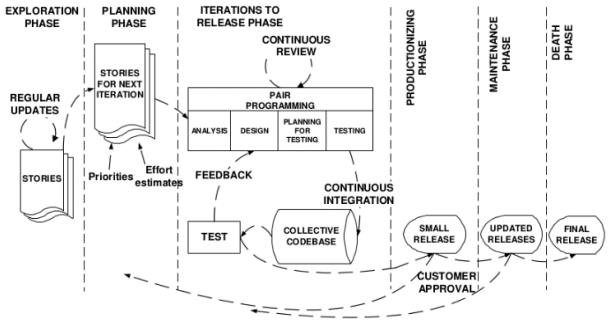
Pada dasarnya, Ren'Py hanyalah sebuah *engine* yang membantu pembentukan *game* serta memudahkan penulisan program (Ren'Py, 2015). Namun untuk menulis program tersebut, Ren'Py masih memerlukan program lain seperti *teks editor* Editra. Editra adalah *editor teks multi-platform* dengan implementasi yang berfokus pada menciptakan antarmuka yang mudah digunakan dan fitur yang membantu dalam pengembangan kode. Saat ini Editra mendukung sintaks dan berbagai fitur lain yang berguna untuk lebih dari 60 bahasa pemrograman. Editra tersedia secara bebas untuk digunakan di bawah Lisensi wxWindows. Saat ini proyek tersebut dalam tahap pengembangan *alpha* tetapi *test builds* dari titik "stabil" sudah tersedia untuk di-*download* dan dicoba pada sistem operasi Windows dan Mac OSX (Universal), saat ini sistem lain yang berbasis Unix dan Linux harus menginstal dari sumber menggunakan *setup script* yang telah disediakan (Editra, 2015).

2.7. Extreme Programming (XP)

XP adalah salah satu cara dalam mengembangkan sebuah *software* yang ringan, efisien, berisiko rendah, fleksibel, dapat diprediksi, cara ilmiah. Yang membedakan cara ini dengan metodologi lain adalah (Beck, 1999) :

1. Cepat, konkrit, dan umpan balik yang berkelanjutan dari sebuah siklus yang pendek.
2. Perencanaan yang bertahap yang dengan cepat datang dengan rencana keseluruhan yang diharapkan berkembang melalui kehidupan sebuah *project*.
3. Kemampuan membuat jadwal yang fleksibel pada implementasi dari fungsionalitas, dan merespon perubahan pada keperluan bisnis.
4. Ketergantungan pada tes secara otomatis yang ditulis oleh *programmer* dan *customer* untuk memantau kemajuan perkembangan, agar memungkinkan sistem untuk berkembang dan menangkap *defect* atau cacat program secara cepat.
5. Ketergantungan pada komunikasi lisan, *test*, dan *source code* untuk berkomunikasi pada struktur sistem.
6. Ketergantungan pada proses desain yang berevolusi sehingga dapat bertahan sepanjang sistem tersebut bertahan.
7. Ketergantungan pada kerjasama yang erat dari programmer dan kemampuan normal.
8. Ketergantungan pada latihan yang bekerja dengan baik dalam jangka pendek, naluri dari *programmer*, dan jangka panjang dari ketertarikan terhadap *project.*

Dalam metode *Extreme Programming* terdapat beberapa tahap pengembangan yang diilustrasikan dalam gambar berikut ini (Beck, 1999);



Gambar 3. Tahapan *Extreme Programming (Candra, 2012)*

1. Tahap Explorasi (Exploration Phase)

Pada tahap ini, *peneliti* menuliskan kebutuhan-kebutuhan dari sistem yang paling mendasar untuk mengumpulkan kebutuhan yang nantinya akan diimplementasikan pada suatu sistem. Informasi yang dikumpulkan berdasarkan pada kebutuhan dasar pada suatu game, serta informasi mengenai aksara Lampung.

2. Tahap Perencanaan (Planning Phase)

Tahap perencanaan berorientasi pada tahap eksplorasi. Pada Tahap ini *peneliti* memperkirakan

kebutuhan *user*, kebutuhan operasi dan kebutuhan system. Maka pada tahap ini akan dihasilkan jadwal pelaksanaan proyek. Rancangan jadwal kegiatan yang dibuat memiliki tujuan untuk memberikan gambaran waktu pelaksanaan pembangunan sistem. Penentuan waktu pembangunan sistem yang terjadwal dimaksudkan untuk dijadikan batasan waktu dalam setiap tahapan proses pembangunannya.

3. Tahap Iterasi (Iterations Phase)

Proses iterasi yang dilakukan pada pembuatan *Game* Edukasi Ajo dan Atu – Belajar Bahasa Lampung ini mengacu pada Gambar 3. Iterasi terdiri dari tiga tahapan utama yaitu Analisis Sistem, Desain Sistem dan Pengujian Sistem.

Analisis

Tahap analisis merupakan tahap penting yang harus dilakukan sebelum memulai perancangan dan pembangunan sistem. Tahap analisis meliputi beberapa aspek, seperti menganalisis kebutuhan dari sistem, baik kebutuhan fungsional dan kebutuhan nonfungsional. Kebutuhan fungsional merupakan kebutuhan mengenai bagaimana suatu sistem melakukan proses kerjanya dan apa saja yang dihasilkan pada proses kerja yang dilakukan oleh sistem. Sedangkan kebutuhan nonfungsional adalah kebutuhan perangkat keras dan perangkat lunak yang digunakan oleh sistem.

Desain

Tahap ini dibuat dengan tujuan untuk memberikan gambaran dari *game* yang akan dibuat. Desain yang dilakukan pada tahap ini meliputi desain karakter dan desain *interface* dari *game* yang akan dibuat.

Pengujian

Pengujian ini meliputi pengujian fungsional dari *game* dan pengujian terhadap hal-hal yang terkait dengan *game* secara teknis. Hasil pengujian dianggap sebagai *feedback* dan jika terdapat hasil yang tidak sesuai akan segera diperbaiki pada iterasi selanjutnya.

4. *Tahap Produksi (Production Phase)*

Tahap ini merupakan tahap dimana *release* akhir dari pembangunan *game*. Bentuk *release* akhir yang dimaksudkan adalah *peneliti* menunjukkan keseluruhan *game* yang telah dibuat kepada user. Pada tahap ini juga akan terjadi pembaharuan sistem berdasarkan hasil yang didapat pada tahap pengujian akhir. Semua saran yang ada pada tahap pengujian diharapkan telah direalisasikan dalam bentuk perubahan sistem.

5. Tahap Pemeliharaan (Maintenance Phase)

Pada tahap ini dilakukan pengecekan akhir pada *game*. Di tahap ini juga dipastikan bahwa implementasi kebutuhan fungsional dan nonfungsional sistem telah terpenuhi. Jika syarat tersebut telah terpenuhi, maka *game* yang akan dibuat dapat dilanjutkan ke tahap publikasi.

6. *Tahap Publikasi Sistem (Death Phase)*

Tahapan ini merupakan tahapan akhir dalam pembangunan sistem yang telah diuji kemudian diimplementasikan sesuai dengan kebutuhan target. *Game* yang dihasilkan dalam penelitian ini akan disebarkan melalui media internet sehingga masyarakat dapat mengunduh *game* tersebut.

3. Multimedia

Multimedia adalah kombinasi dari teks, gambar, suara, animasi, dan video yang dikirimkan kepada pengguna oleh komputer atau elektronik lain atau cara manipulasi secara digital (Vaughan, 1993). Teknologi multimedia sangat diperlukan dalam pengembangan *game* edukasi.

3. HASIL DAN PEMBAHASAN

3.1. Tahap Iterasi *(Iterations Phase)*.

Pada tahap ini telah melewati iterasi kedua. Pada iterasi kedua, desain mengalami beberapa perubahan antara lain:

1. Karakter

Pada iterasi kedua, desain karakter mengalami perubahan pada pakaian karakter. Desain pakaian karakter utama menggunakan motif tapis untuk memperlihatkan budaya Lampung.

Gambar 4. Perubahan desain karakter utama

Selain itu, ada beberapa tambahan karakter seperti orang tua karakter utama.

Gambar 5. Orang tua dari karakter utama

2. Interface

Pada tahap ini, *interface* yang telah dirancang pada tahap desain, telah diimplementasikan ke dalam *game*. Dalam proses implementasinya, *interface* tidak mengalami perubahan dari desain awal. Pada implementasinya, *coding interface* berada pada *file screens.rpy* yang sengaja dibuat untuk menampung seluruh kode pemrograman *interface* pada *game*.

Gambar 6. *Interface* yang telah diterapkan kedalam *game*

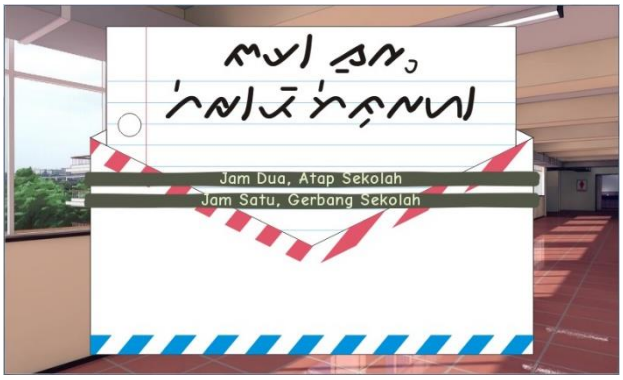
3. Background

Pada desain *background*, *peneliti* menggunakan desain *background* yang telah disediakan dan bebas digunakan dari pihak Ren'Py. Desain karakter diunduh dari forum khusus *developer* yang menggunakan *engine* Ren'Py. Desain *background* yang diunduh berjumlah 35 *items* yang terdiri dari beberapa tempat dari ruangan-ruangan kelas, sekolah, beserta beberapa bagian diluar sekolah seperti tengah kota, jalanan, dan bangunan-bangunan.

Gambar 4.4. Salah satu desain *background* yang digunakan di dalam *game*

4. Jalur Cerita *(Storyline)*

Pada *game* "Belajar Aksara Lampung ", jalur cerita memiliki percabangan cerita yang akan menuju pada 4 jenis *ending*. Tiap *ending* ditentukan dari jalur cerita yang dipilih oleh pemain dan total poin yang dikumpulkan oleh pemain selama permainan berlangsung. Poin didapatkan dari jawaban yang dipilih oleh pemain. Tiap jawaban benar maka akan sistem akan secara otomatis poin akan diakumulasikan sehingga didapatkan jumlah akhir yang akan menentukan arah *ending* permainan. Dalam permainan, terdapat 6 pilihan (*choice*) pada cerita. Dimana tiap bagian pilihan terdapat dua buah *choice*. Salah satu dari kedua *choice* adalah pilihan yang benar dan memiliki poin sebesar 5. Dengan begitu, pada akhir *game*, poin terbesar yang dapat dikumpulkan pemain adalah 30.



Gambar 7. Salah satu dari 6 *choice* yang ada dari *game*

Sama seperti jumlah *ending* yang ada, tiap *ending* pada *game* ini memiliki tipe yang berbeda-beda. Pada sistem *multi-ending* di *game* yang diterapkan, *ending* pertama harus dicapai dengan poin 30 yang berarti setiap pemain memilih *choice* harus semuanya tepat. Pada *ending* ini, Papa dari Ajo dan Atu akan pulang dan mengajak liburan ke objek wisata Pulau Pahawang. Tipe *ending* pertama ini adalah *Great Ending*.

Pada *ending* kedua, harus dicapai dengan poin 25 yang berarti 5 dari 6 *choice* yang dipilih harus benar. Pada *ending* ini, Papa dari Ajo dan Atu akan pulang dan mengajak liburan ke objek wisata Way Kambas. Tipe *ending* kedua ini adalah *Good Ending*.

Pada *ending* ketiga, harus dicapai dengan poin 20 yang berarti 4 dari 6 *choice* yang dipilih harus benar. Pada *ending* ini, Papa dari Ajo dan Atu akan pulang dan akan mengajak mengunjungi objek wisata Menara Siger. Tipe *ending* ketiga ini adalah *Normal Ending*.

Pada *ending* keempat, merupakan *ending* yang akan didapatkan pemain jika poin kurang dari atau sama dengan 15. Pada ending ini, Ajo dan Atu mengetahui identitas pengirim surat dan Papa dari Ajo dan Atu tidak akan pulang dalam jangka waktu 3 minggu. Tipe *ending* keempat ini adalah *Bad Ending*.

5. Pengujian

Pada tahap ini, peneliti melakukan pengujian ulang untuk memastikan tidak adanya *bugs* pada *game* setelah semua *assets* baru diaplikasikan kedalam *game*. Pengujian ini meliputi pengujian fungsional dari *game* dan pengujian terhadap hal-hal yang terkait dengan *game* secara teknis. Hasil pengujian dianggap sebagai *feedback* dan jika

terdapat hasil yang tidak sesuai segera diperbaiki pada iterasi selanjutnya. Pengujian dilakukan pada beberapa perangkat, diantaranya yaitu Bluestack Android *Emulator*, Samsung Galaxy SL i9003, dan Nokia XL Dual SIM RM-1042. Beberapa pengujian yang dilakukan yaitu antara lain:

Tabel 1. Pengujian ulang pada *game*

No.	Pengujian	Hasil yang Diharapkan
1.	Menu pada *Startscreen*	Menu dapat berfungsi dengan baik
2.	*Asset*	Seluruh *Asset* dapat terlihat saat permainan mulai tanpa adanya kerusakan seperti gambar hilang, kontras warna berubah dan terpotong.
3.	*Imagemaps*	*Dots* pada *imagemaps* berada pada posisi yang seharusnya.
4.	*Choice*	Percabangan cerita dapat jump ke label yang seharusnya.
5.	*Quickmenu*	Fungsi *save*, *load* dan *auto* dan *skip* dapat berfungsi dengan baik.
6.	*Point*	*Point* dapat menampilkan nilai yang sesuai dengan *choice* yang dipilih oleh pengguna.

3.2. Tahap Produksi (*Production Phase*)

Tahap ini merupakan tahap dimana *release* akhir dari pembangunan *game*. Bentuk *release* akhir yang dimaksudkan adalah *peneliti* menunjukkan keseluruhan *game* yang telah dibuat kepada target. Pada tahap ini juga terjadi pembaharuan sistem berdasarkan hasil yang didapat pada tahap pengujian akhir. Semua saran yang ada pada tahap pengujian diharapkan telah direalisasikan dalam bentuk perubahan sistem. Pada tahap ini, *Game* Edukasi Belajar Bahasa Lampung ditunjukkan kepada murid dan guru Bahasa Lampung SMP Negeri 1 Bandar Lampung sebagai *game* tester atau penguji *game*. Selanjutnya, *peneliti* meminta *Game* Tester untuk memberikan saran kepada *game* yang diuji. Semua saran diharapkan dapat direalisasikan di tahap berikutnya.

Gambar 8. Peserta *Small-Release* SMP Negeri 1 BDL

Gambar 9. Siswa SMP Negeri 1 sedang mencoba *game* edukasi Belajar Aksara Lampung

Gambar 10. Tim peneliti sedang menjelaskan cara penggunaan *game* edukasi Belajar Aksara Lampung

Setelah tim peneliti melakukan *small release* di SMP Negeri 1 Bandar Lampung , maka didapatkan beberapa hasil review. Hasil *review* yang didapatkan tersebut berupa saran dan penemuan *bugs*.

Tabel 2. Saran yang didapatkan setelah dilakukannya *small-release*

No.	Saran
1	*Size* karakter terlalu kecil. Tidak sesuai dengan besar layar
2	Pada bagian Anak Surat, Sebaiknya ditambahkan penjelasan tiap posisi pembacaan anak surat
3	Jumlah *choice* ditambahkan
4	Ukuran *font* sedikit lebih diperbesar
5	Pada *Startmenu* sebaiknya dipasang gambar ikon Lampung.

Selain saran, pada saat melakukan *small-release*, *peneliti* menemukan dua buah *bugs* pada game. Berikut daftar *bugs* yang ditemukan:

Tabel 3. *Bugs* yang didapatkan setelah dilakukannya *small-release*

No.	*Bugs*
1	Percabangan cerita nomor 4 mengalami *error*.
2	Poin pada *choice* nomor 4 tidak ada sehingga poin tidak bertambah walaupun memilih jawaban benar

Semua saran dan *bugs* yang didapatkan *peneliti* dari hasil *small-release* ditinjau, diperbaiki dan segera diterapkan ke dalam *game*. Setelah memasuki tahap produksi dan mendapatkan hasil, *peneliti* mulai memasuki Tahap Pemeliharaan atau *Maintenance Phase*.

3.3. Tahap Pemeliharaan (*Maintenance Phase*)

Pada tahap ini pengembangan melakukan peninjauan dan perubahan kembali setelah melakukan *small-release*. Pada tahap Produksi sebelumnya, *peneliti* mendapatkan beberapa saran dan *bugs* yang diperbaiki di tahap ini. Seluruh saran dan *bugs* terdapat pada tabel 2. dan tabel 3. Beberapa perubahan yang terjadi setelah semua saran dan *bugs* tersebut diterapkan dan diperbaiki yaitu :

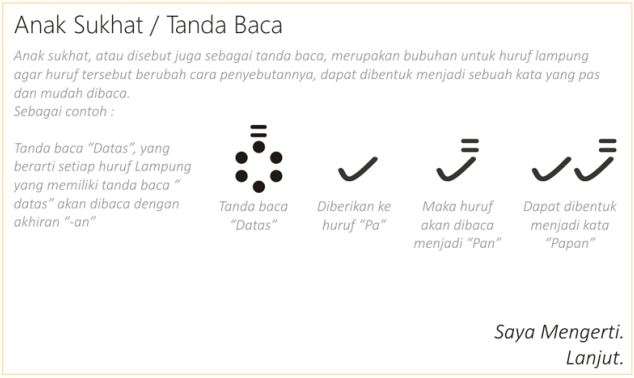
1. *Size karakter*

Ukuran karakter sebelum dilakukan perubahan yaitu memiliki tinggi 450 *pixel*. Perubahan dilakukan dengan menggunakan *software* Adobe Photoshop CS5. Setelah dilakukan perubahan, ukuran tinggi karakter menjadi 500 *pixel*.

Gambar 11. Perbandingan ukuran karakter dengan tinggi 450 *pixels* (kiri) dan 500 *pixels* (kanan)

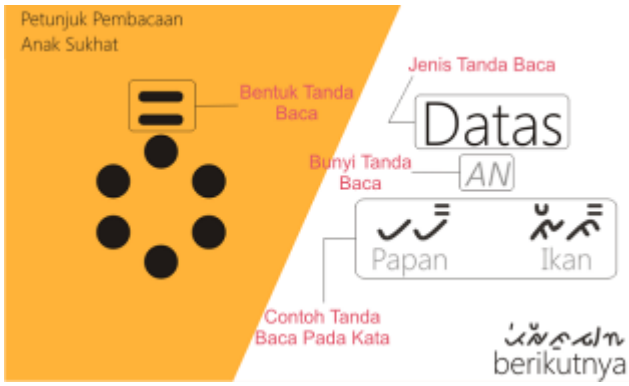
2. *Petunjuk anak surat*

Pada bagian anak surat, awalnya telah memiliki petunjuk cara membaca anak surat. Namun setelah dilakukan *small-release*, didapatkan bahwa murid masih sedikit kesulitan dan belum benar-benar mengerti cara membaca anak surat tersebut.



Gambar 12. Cara membaca anak surat

Dari hasil tersebut, *peneliti* memutuskan untuk menambahkan petunjuk berupa penjelasan lebih detil cara pembacaan anak surat. Petunjuk baru ini dibuat dengan menggunakan *software* CorelDraw X4. Petunjuk baru diletakkan tepat setelah petunjuk cara membaca anak surat seperti yang ditunjukkan pada gambar 13.



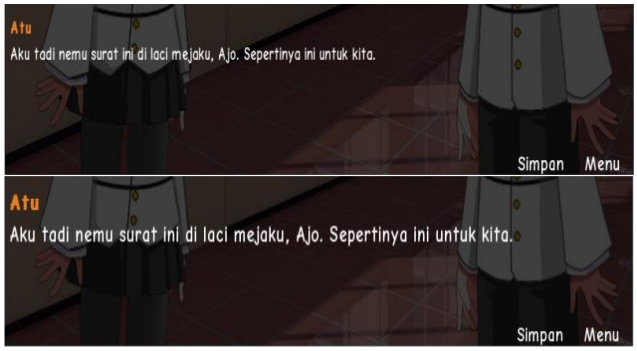
Gambar 13. Petunjuk pembacaan anak surat dengan deskripsi penjelasan

3. Jumlah Pilihan *(Choice)*

Pilihan atau *choice* yang diterapkan sebelum dilakukannya *small-release* berjumlah 4 pilihan. Namun setelah dilakukannya *small-release*, didapatkan hasil bahwa 4 pilihan terlalu sedikit. Maka *peneliti* beserta murid memutuskan untuk menambahkan *choice* menjadi 6. Awal total *point* terbesar berjumlah 20. Dengan dirubahnya jumlah *choice* maka jumlah total poin terbesar juga berubah menjadi 30.

4. *Ukuran Font (Font Size)*

Sebelum dilakukannya *small-release*, ukuran font dengan *size* 15 memiliki masalah berupa terlalu kecilnya tulisan pada layar *smartphone* dengan ukuran 480x800 *pixel*. Maka *peneliti* melakukan perubahan pada *font size* yang awalnya 15 menjadi 20.



Gambar 14. Perbandingan antara *Text size* 15 (atas) dengan *Text size* 20 (bawah)

5. *Startscreen*

Sebelumnya, *game* ini hanya menggunakan warna hijau pastel polos (*code hex* = #00cc99) yang merupakan warna dari *template* yang telah disediakan oleh engine Ren'Py. Setelah dilakukannya *small-release*, didapatkan bahwa *startscreen* sebaiknya dipasangkan gambar ikon Lampung. Maka dari itu *peneliti* menggunakan gambar Menara Siger pada *startscreen*.

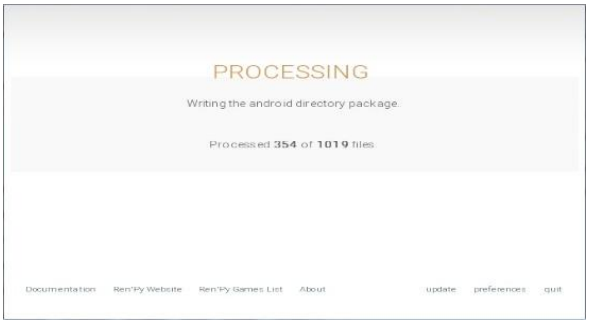
Gambar 15. *Startscreen* menggunakan gambar Menara Siger

6. *Percabangan Cerita*

Pada saat dilakukannya *small-release*, ditemukannya *bugs* pada percabangan cerita nomor 4. *Bugs* ini membuat *game* yang sedang dijalankan akan mengalami pesan *error*. Hal ini dikarenakan pada saat melakukan percabangan cerita, sistem harus mencari kode tujuan dari perintah *jump*. *Error* terjadi karena tidak ditemukannya label dari tujuan perintah *jump*. *Peneliti* telah memperbaiki label tujuan dari perintah *jump*. Dengan begitu tujuan percabangan cerita nomor 4 dapat berjalan dengan baik. Selain adanya *error* pada percabangan cerita, ada *bugs* lain yang terdapat pada percabangan cerita nomor 4. Percabangan cerita selalu terjadi pada menu *choice* sehingga jika pemain memilih salah satu *choice*, pemain akan diarahkan ke alur cerita yang sesuai dengan pilihan sebelumnya. Pada tiap *choice* tersebut, salah satu nya terdapat jawaban yang benar dan point akan terakumulasi karena adanya perintah *"$ point +=5"* pada jawaban yang benar. Namun, pada percabangan cerita nomor 4, *peneliti* tidak memasukkan perintah *"$ point +=5"* sehingga poin tidak terakumulasi sebagaimana mestinya. Setelah menemukan dimana letak kesalahan, *peneliti* menambahkan perintah tersebut kedalam salah satu *choice* yang benar sehingga point dapat terakumulasi.

7. *Test Aplikasi*

Pada tahap *maintenance*, *peneliti* juga diharuskan untuk mengetes langsung *game* yang telah dibuat ke dalam perangkat Android. Pada kali ini *peneliti* menggunakan Bluestack Android *emulator* untuk *testing game* yang telah di-*build* dengan menggunakan *engine* Ren'Py.



Gambar 16. Proses *build game* pada *engine* Ren'Py

Setelah proses *build* selesai, *peneliti* melakukan proses *install* pada Bluestack Android *emulator*. *File* hasil *build* dari *engine* Ren'Py memiliki ekstensi *.apk*. Dengan adanya Bluestack Android *emulator* yang telah ter*install*, *Peneliti* cukup mengklik file *.apk* tersebut 2 kali dan aplikasi terinstall secara otomatis pada Bluestack.

Gambar 17. Bluestack Android *emulator* melakukan proses instalasi aplikasi

Gambar 18. Proses instalasi *game* pada Bluestack berhasil

3.4. Tahap Publikasi Sistem (*Death Phase*)

Pada tahap publikasi atau *death phase*, *game* yang telah dibuat sudah selesai dan siap untuk memasuki tahap *release* atau yang biasa disebut dengan *"Master Up"*. Di tahap ini, *game* sudah tidak ada lagi yang harus diperbaiki karena *game* yang telah *release* sudah sempurna dan telah melalui proses pengujian sehingga tidak ditemukan lagi adanya *bugs* dan *error*. Untuk melakukan *release*, *peneliti* menggunakan *platform Google-drive*. Dengan menggunakan *Google-drive*, *peneliti* tidak memerlukan mengeluarkan biaya untuk melakukan *release* seperti pada *Play Store* milik Google. *Peneliti* juga dapat memuat lebih banyak informasi mengenai *game* yang dibuat. Jika *peneliti* memutuskan untuk mengembangkan *sequel* dari *game* yang dibuat, *peneliti* dapat melakukan *release* di tempat yang sama. Untuk proses *upload* file *game* yang memiliki ekstensi .apk.

4. KESIMPULAN DAN SARAN

4.1. Kesimpulan

Berdasarkan hasil perancangan dan implementasi *Game* Edukasi Belajar Aksara Lampung ini dapat ditarik kesimpulan sebagai berikut:

1. *Game* Edukasi yang dibangun ini dapat digunakan untuk media pembelajaran Aksara Lampung.
2. Menurut hasil survey pada saat *small-release*, 80% dari seluruh peserta survey menunjukkan bahwa *game* edukasi ini memiliki tampilan yang simpel dan mudah dimengerti oleh pengguna.
3. *Game* Edukasi dapat dimainkan dimanapun dikarenakan *game* berekstensi .APK yang dapat di install di perangkat yang memiliki OS Android versi 2.3+.
4. Ada nya sistem *Multi-Ending* membuat pengguna lebih tertarik untuk terus mencoba mencari *ending* yang lain sehingga pengguna dapat lebih mudah belajar karena mengulang permainan berkali-kali.

4.2. Saran

Berdasarkan pengalaman selama berlangsungnya perancangan dan implementasi *game* Edukasi Belajar Aksara Lampung ini terdapat beberapa saran sebagai berikut:

Perbanyak *choice* pada setiap jalur cerita sehingga permainan menjadi lebih menantang.

Melakukan release di *Play Store* sehingga *game* yang dibuat menjadi lebih dikenal dan lebih mudah *didownload*.

Perbanyak varian *Ending* sehingga pengguna dapat melakukan lebih banyak eksplorasi cerita yang akan membuat pengguna lebih mudah belajar aksara Lampung.

4. Lebih perbanyak tentang sejarah dan pariwisata Lampung sehingga lebih mengenalkan Lampung pada pengguna.
5. Saat ini, *Game* edukasi yang dibuat menggunakan *layout interface* yang disediakan oleh Ren'Py. Menggunakan *Custom Interface* pada *game* dapat membuat tampilan menjadi lebih menarik.

5. DAFTAR PUSTAKA

ARYANTIO, A. & MUNIR, R. 2015. Pengenalan Aksara Lampung Menggunakan Jaringan Syaraf Tiruan. Makalah KNIF, 34-38.

BECK, K. 1999. *Extreme Programming Explained: Embrace Change.* (First Edition, ISBN 0201616416), Addison-Wesley Longman Publishing Co., Inc., Hal. 9.

CANDRA, B.A. 2012. Rancang Bangun Sistem Informasi Manajemen Terpadu (SIMANTEP) Online PT. PLN (Persero) Sektor Pembangkitan Tarahan Lampung Dengan Metode *Extreme Programming*. Jurnal Komputasi, Vol 1, No. 1, 31-24.

DESPA, D. & KURNIAWAN, A. & KOMARUDIN, M. & MARDIANA & NAMA, G. F. 2015. *Smart monitoring of electrical quantities based on single board computer BCM2835. 2nd International Conference on Information Technology, Computer, and Electrical Engineering*, pp 315-320, Indonesia.

EDITRA 2015. *Welcome to* Editra.org, http://www.editra.org/.

EGENFELDT, N. S. 2015. *Making sweet music: The Educational Use of Computer Games*. Url:http://www.egenfeldt.eu/papers/sweet_ music.pdf, diakses 16 Desember 2015.

KIILI, K. 2004. *Digital game-based learning: Towards an experiental gaming model. Internet and Higher Education* 8, 13-24.

NAMA, G. F. & ULVAN, M & ULVAN, A. & HANAFI, A. M. 2015. *Design and Implementation of Web-Based Geographic Information System for Public Services in Bandar Lampung City – Indonesia*. *International Conference on Science in Information Technology* (ICSITech), pp 270-275, Yogyakarta, Indonesia.

NAMA, G. F. & KOMARUDIN, M. & SEPTAMA, H. D. 2015. *Performance Analysis of ArubaTM Wireless Local Area Network Lampung University*. *International Conference on Science in Information Technology* (ICSITech), pp 41-46, Yogyakarta, Indonesia.

NAMA, G. F. & KOMARUDIN, M. & PRIAMBODO, H. & MARDIANA & SEPTAMA, H. D. 2014. *Electricity, Temperature, and Network Utilization Monitoring at Lampung University Data Centre Using Low Cost Low Power Single Board Mini Computer. Regional Conference On Computer Information Engineering*, Indonesia, pp. 184-189.

REN'PY. 2015. Why Ren'Py?. http://www.renpy.org/why.html.

VAUGHAN, T. 1993. *Multimedia: Making It Work*. (first edition, ISBN 0-07-881869-9), Osborne/McGraw-Hill, Berkeley, Hal. 3.

Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) p-ISSN: 2355-7699
Vol. 3, No. 4, Desember 2016, hlm. 248-252 e-ISSN: 2528-6579

SISTEM PENILAIAN OTOMATIS JAWABAN ESAI PADA ELEARNING BELAJARDISINI.COM

Eko Sakti Pramukantoro[1]

[1]Fakultas Ilmu komputer Universitas Brawijaya
Email: [1]ekosakti@ub.ac.id

(Naskah masuk: 1 Oktober 2016, diterima untuk diterbitkan: 26 Desember 2016)

Abstrak

Belajardisini.com adalah sebuah *e-learning* berbasis *gamification*. *S*istem tersebut menggunakan proses manual untuk melakukan koreksi jawaban uraian pendek(esai). Bertambahnya jumlah pelajar dan banyaknya ujian mengakibatkan pengajar harus meluangkan waktu untuk menilai jawaban soal ujian, sehingga penggunaan e-*learning* dan tanpa e-*learning* dari sisi efisiensi waktu tidak ada perbedaan. Oleh karena itu diperlukan sistem penilaian otomatis pada *e-learning* untuk jawaban esai. Sistem (*automated essay scooring*)AES dengan algorima *Cosine Similarity* dikembangkan untuk menjawab kebutuhan tersebut. Algoritma cosine dipilih atas hasil penelitian sebelumnya yang membandingkan antara Cosine dan LSA, dimana Cosine lebih unggul dari LSA dari segi komputasi. Sistem yang diusulkan diintegrasikan ke dalam *e-learning*, sehingga perlu analisis performa sistem. Untuk mengetahui performa sistem dilakukan uji berdasarkan penggunaan CPU, Memori dan *page load time*. Proses uji performa *server* dari dua jenis soal yang diujikan disimpulkan bahwa semakin besar pengguna yang mengakses sistem maka semakin besar juga *CPU Usage* yang dibutuhkan yaitu paling besar adalah 0,2556 %. Untuk pengujian *page load time* dan *memory usage* tidak ditemukan perbedaan yang cukup signifikan ketika sistem digunakan oleh satu pengguna maupun banyak pengguna yaitu antara 0,208392 detik sampai 0,406842 detik untuk *page load time* dan antara 1,49 % sampai 1,56 % untuk *memory usage*.

Kata kunci: *automated essay scooring, cosine similarity, elearning*

Abstract

Belajardisini.com is elearning based on gamification concept. In that system for correction the answer of essay exam is still done manually. Increasing the number of students and the number of exam makes teachers having to take time to assess the answers to the exam, so using elearning and without elearning in asessment there is no different esspisially in term time efficiency. Therefore we need an automatic scoring system for the essay. Automated essay scooring system using cosine similarity is developed to answer those needs. Cosine algorithm is selected based on the results of previous studies that comparing cosine and LSA, where cosine superior in terms of computation. The proposed system is integrated into the e-learning, so analysis of system performance is required. Analysis system performance based on the use of CPU, memory and page load time. The result shown server performance of the two types of questions that tested concluded that the users accessing the system, the greater the required CPU Usage is at most 0.2556%. For the test page load time and memory usage found no significant difference when the system is used by a single user or multiple users is between 0.208392 seconds to 0.406842 seconds to page load time and between 1.49% to 1.56% for memory usage.

Keywords: *automated essay scooring, cosine similarity, elearning*

1. PENDAHULUAN

Dalam bidang pendidikan, situs web bisa digunakan sebagai sarana pembelajaran antara dosen dan mahasiswa yang disebut *elearning*. Pada penelitian sebelumnya dihasilkan sebuah media pembelajaran berbasis *gamification*, yaitu memadukan konsep *game* dalam sebuah *elearning* (Sakti, 2015). Dalam *elearning* tersebut tugas yang diberikan oleh dosen dapat dikerjakan oleh mahasiswa secara online. Tipe tugas yang ditawarkan dalam media belajar tersebut dapat dikategorikan menjadi dua jenis soal pilihan ganda dan soal uraian pendek atau esai.

Dalam perkembangannya fitur ujian esai menjadi kendala, karena bertambahnya jumlah pelajar dan banyaknya ujian mengakibatkan pengajar atau dosen harus meluangkan waktu untuk menilai jawaban soal ujian. Pada umumnya sistem koreksi jawaban ujian esai dilakukan secara manual, sehingga penggunaan e-*learning* dan tanpa e-*learning* dari sisi efisiensi waktu tidak ada perbedaan. Oleh karena itu dibutuhkan sistem penilaian otomatis pada ujian esai untuk mempercepat proses penilaian jawaban mahasiswa, akan tetapi tidak boleh mengesampingkan akurasi penilaian.

Ujian esai berbeda dengan ujian pilihan ganda yang hanya ada benar dan salah. Penilaian pada ujian esai lebih sulit karena harus menganalisis dan

mengerti teks yang ditulis mahasiswa. Ketika penilaian otomatis diterapkan pada e*leaning* ada faktor yang tak kalah penting, yaitu performa server dengan sumber daya yang dibutuhkan. Perlu sebuah sistem penilaian otomatis dengan komputasi ringan. Hal ini yang menjadi kebutuhan khusus terhadap pengembangan sistem penilaian otomatis. Permasalahan tersebut dijawab dalam penelitan (Eko Sakti, 2016) dengan membandingkan algoritma *string similarity* (cosine) dan *corpus based* (*latent semantic analysis*). Dari penelitian tersebut didapat kesimpulan kedua algoritma mempunyai akurasi penilaian yang kompetitif akan tetapi cosine memiliki performa komputasi yang lebih baik dibandingkan dengan LSA, sehingga *cosine similarity* yang diusulkan untuk diterapkan pada AES.

Pada penelitian ini dilakukan integrasi sistem AES berdasarkan algorima cosine similarity dan dilakukan pengujian untuk mengetahui performa sistem AES yang telah dibangun. Parameter uji performa sistem berdasarkan penggunaan CPU, memori dan page load. Sehingga dengan adanya sistem AES dapat membantu dosen atau pengajar dalam proses koreksi jawaban esai.

2. TEORI

2.1 *Automated essay scooring*

Automated essay scooring (AES) adalah sebuah sistem penilaian otomatis dengan cara membandingkan dua jawaban kemudian dikalkulasi hasil dari perbandingan menggunakan metode-metode yang ada. Contoh metode yang bisa digunakan untuk penerapan AES adalah *Corpus based similarity* dan *String based similarity*. *Corpus Based Similarity* (CBS) adalah metode yang mengukur kesamaan arti dari setiap kata berdasarkan sebuah kamus yang telah disediakan (Gomma, 2012). *Latent Semantic Analysis* (LSA) merupakan teknik yang popular dari CBS. *String Based Similarity* (CBS) adalah metode yang mengukur kesamaan *string* antara dua text tanpa melihat arti dari text tersebut. Teknik yang bisa digunakan untuk CBS adalah *Cosine Similarity*, *Dice* dan *Jaccard*.

2.2 Cosine Similarity

Cosine similarity merupakan metode fungsi pengukuran kesamaan antara dua vektor yang berbeda dengan mengukur cosinus dari sudut antara mereka (Kurniawan, 2014).

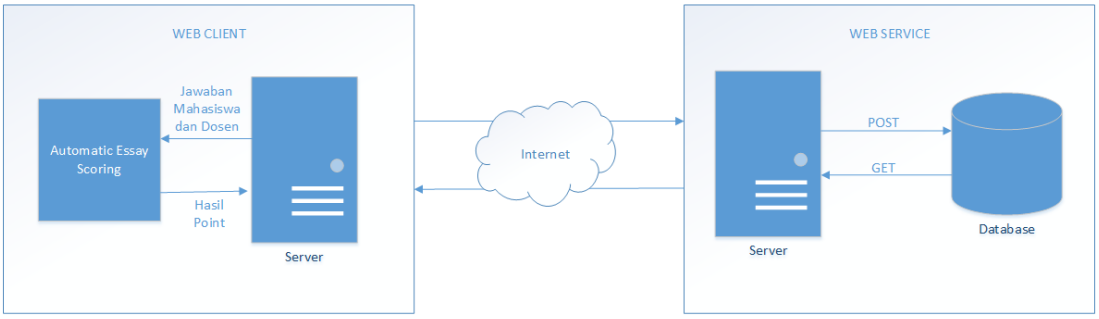
Metode ini dapat digunakan untuk melihat seberapa miripkah sebuah kalimat antara dua dokumen. Semakin besar nilai cosinus (maksimal 1) maka semakin mirip dokumen yang dibandingkan. Nilai cosinus 1 menyatakan kemiripan 100% sedangkan jika nilai 0 ketidakmiripannya 100%.

3. Sistem AES

Elearning yang telah dikembangkan berjalan diatas mesin dengan spesifikasi 1 GB *Memory*, 30 GB *Disk*, 2 TB *Transfer* dan dapat diakses melalui domain http://belajardisini.com. Aplikasi tersebut berbasis website dan dapat diakses oleh pengguna melalui sebauah aplikasi browser.

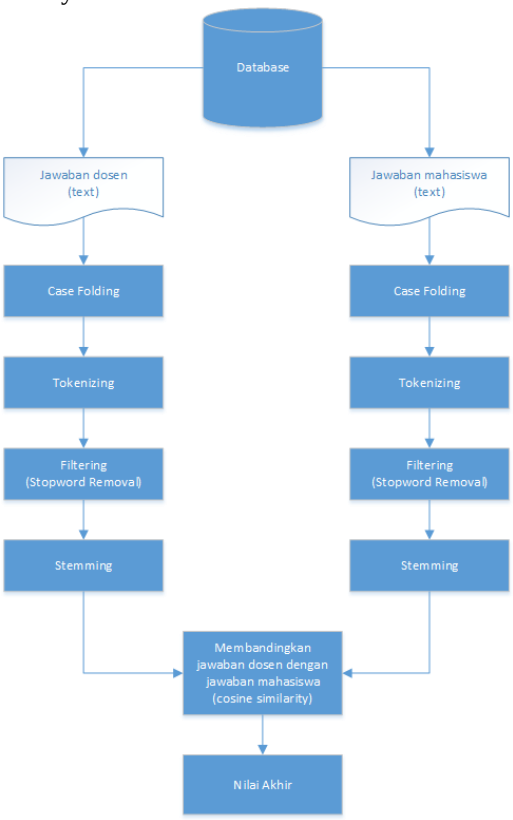
Untuk aplikasi sistem penilaian otomatis jawaban uraian pendek pada *elearning* ini dibagi atas dua sistem yaitu *web service* dan *web client*. Kedua sistem tersebut berkomunikasi dengan menggunakan *Web Service*. Dosen dapat melihat mata kuliah diajar dan dapat mengelola *challenge* uraian pendek yang nantinya bisa dikerjakan oleh mahasiswa. Semua proses data yang dilakukan pada *web client* nantinya akan dikirim ke *web service* dan disimpan ke *database*.

Gambar 1, menunjukan terdapat dua sistem utama yaitu *web client* dan *web service*. *Web service* berguna mengambil data jawaban dosen dan mahasiswa pada *database* kemudian dikirimkan ke *web client*. Setelah data jawaban diterima oleh *web client* kemudian akan masuk ke proses *Automatic Essay Scoring* (AES). Proses AES diletakkan pada *web client* dikarenakan proses AES hanya membutuhkan data jawaban yang diproses pada halaman detail jawaban mahasiswa. Data jawaban nantinya akan diproses pada AES dengan menggunakan metode *cosine similarity*. Jawaban dosen dan jawaban mahasiswa dibandingkan kemudian akan muncul hasil *point* berdasarkan perhitungan *cosine similarity*.



Gambar 1. Arsitektur komunikasi sistem AES

Proses pengoreksian otomatis pada dosen dapat dijabarkan lagi menjadi beberapa proses. Pertama jawaban dari dosen dan mahasiswa harus melalui text processing untuk menghilangkan kata-kata yang tidak memiliki arti penting kemudian baru masuk ke proses perhitungan dengan metode cosine similarity.



Gambar 2. Diagram Alur Sistem Penilaian Otomatis

Gambar 2 menjelaskan mengenai alur sistem pada penilaian otomatis jawaban uraian pendek. Jawaban yang sudah dosen inputkan langsung tersimpan di dalam *database*, selanjutnya jawaban tersebut akan diolah oleh sistem. Langkah yang pertama, jawaban tersebut akan mengalami proses *case folding*. Pada *case folding* ini variasi huruf harus diseragamkan dan tanda baca harus dihilangkan untuk menghilangkan *noise* pada saat pengambilan informasi. Selanjutnya langkah kedua, jawaban tersebut akan mengalami *tokenizing*. Proses *tokenizing* ini dilakukan untuk pemecahan kalimat menjadi kata.

Langkah yang ketiga yaitu melakukan *filtering* (*stopward removal*), proses ini dilakukan untuk pembuangan kata yang tidak penting. Dengan dibuangnya *stopwords*, ukuran kosakata menjadi berkurang sehingga hanya kata-kata penting yang terdapat dalam dokumen dan diharapkan akan menjadi kata-kata yang memiliki bobot yang tinggi. Langkah yang terakhir yaitu *stemming*, proses perubahan suatu kata menjadi kata dasar. Setelah melewati pengolahan jawaban seperti yang telah dijelaskan diatas, maka jawaban akan dicocokkan dengan jawaban dari mahasiswa.

Jawaban mahasiswa sebelumnya juga mengalami proses yang sama seperti diatas sebelum akhirnya dicocokkan dengan jawaban dosen. Setelah melakukan pencocokan jawaban dosen dan mahasiswa, maka akan dihitung penilaiannya menggunakan Cosine Similarity. Setelah mendapatkan penilaian, data nilai akan disimpan di database dan juga hasil nilai tersebut akan keluar sebagai output.

Untuk proses perhitungan otomatisnya menggunakan metode *cosine similarity*. *Script cosine similarity* ditunjukkan pada Tabel 1

Tabel 1. *Script cosine similarity*

```
<?php

class CosineSimilarity {

public function similarity(array $vec1,
array $vec2) {
    return $this->_dotProduct($vec1,
$vec2) / ($this->_absVector($vec1) *
$this->_absVector($vec2));
  }

public function _dotProduct(array
$vec1, array $vec2) {
$result = 0;
foreach (array_keys($vec1) as $key1) {
foreach (array_keys($vec2) as $key2) {
if ($key1 === $key2) $result +=
$vec1[$key1] * $vec2[$key2];
}
}
return $result;
}

public function _absVector(array $vec)
{
$result = 0;
foreach (array_values($vec) as $value)
{
$result += $value * $value;
}
return sqrt($result);
}
}
```

4. Pengujian dan pembahasan

Pengujian dilakukan untuk mengetahui performa sistem AES yang telah diintegrasikan dengan aplikasi elearning. Parameter performa dengan melihat seberapa besar *Page load time*, *memory usage* dan CPU *Usage* yang digunakan oleh sistem AES.

Proses pengujian performa server berfungsi untuk melihat seberapa besar pengaruh proses penilaian otomatis terhadap performa server. Proses pengujian performa server dibantu menggunakan *Webserver Stress Tool* yang berfungsi untuk membuat skenario seberapa banyak pengguna yang

mengakses sistem. Terdapat tiga skenario yang digunakan. Skenario pertama hanya ada satu

Tabel 2. Hasil Pengujian *Page Load Time*

JENIS SOAL	Skenario 1	Skenario 2	Skenario 3
Soal A	0,231384	0,228456	0,406842
Soal B	0,234756	0,208392	0,209024

Tabel 3. Hasil Pengujian *Memory Usage*

JENIS SOAL	Skenario 1	Skenario 2	Skenario 3
Soal A	1,5	1,49	1,5
Soal B	1,56	1,5	1,5

Tabel 4. Hasil Pengujian *CPU Usage*

JENIS SOAL	Skenario 1	Skenario 2	Skenario 3
Soal A	0,011	0,1822	0,2556
Soal B	0,0308	0,1484	0,2318

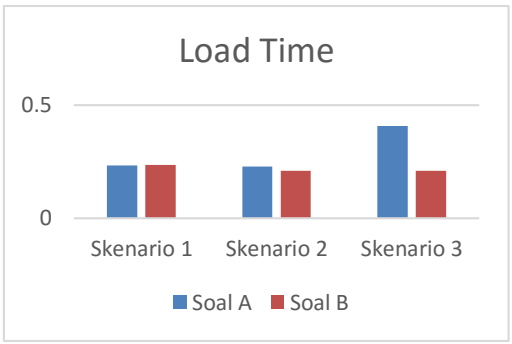
pengguna yang mengakses sistem, skenario kedua 50 pengguna kemudian skenario ketiga 100 pengguna.

Proses pengujian dilakukan terhadap dua jenis soal uraian pendek yaitu soal sebutkan dan apa perbedaan. Performa server yang dihitung adalah *page load time*, *memory usage* dan *CPU usage*. Pengujian dilakukan terhadap 50 jawaban mahasiswa dari masing-masing jenis soal. Dari hasil pengujian yang dilampirkan pada lampiran diambil hasil rata-rata nya kemudian dipisahkan antara hasil *page load time*, *memory usage* dan *CPU usage*.

Berdasarkan kasus uji pada Tabel 2, didapatkan hasil sebagai berikut :

a. Pada skenario 1 rata-rata *Page Load Time* yang dibutuhkan untuk mengakses jenis soal A adalah 0,231384 detik dan jenis soal B adalah 0,234756 detik.
b. Pada skenario 2 rata-rata *Page Load Time* yang dibutuhkan untuk mengakses jenis soal A adalah 0,228456 detik dan jenis soal B adalah 0,208392 detik.
c. Pada skenario 3 rata-rata *Page Load Time* yang dibutuhkan untuk mengakses jenis soal A adalah 0,406842 detik dan jenis soal B adalah 0,209024 detik.

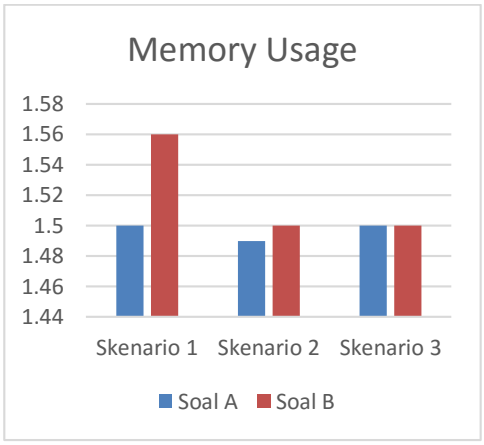
Pada Gambar 3 merupakan grafik dari hasil pengujian *page load time*. Berdasarkan grafik untuk hasil pengujian antara skenario 1 dan skenario 2 tidak mempunyai perbedaan yang signifikan. Untuk skenario 3 perbedaan yang besar terdapat pada pengujian jenis soal A yang membutuhkan waktu lebih banyak dari pada skenario lainnya



Gambar 3. Grafik Hasil Pengujian *Page Load Time*

Berdasarkan kasus uji pada Tabel 3 didapatkan hasil sebagai berikut :

a. Pada skenario 1 rata-rata *Memory Usage* yang dibutuhkan untuk mengakses jenis soal A adalah 1,5 MB dan jenis soal B adalah 1,56 MB.
b. Pada skenario 2 rata-rata *Memory Usage* yang dibutuhkan untuk mengakses jenis soal A adalah 1,49 MB dan jenis soal B adalah 1,5 MB.
c. Pada skenario 3 rata-rata *Memory Usage* yang dibutuhkan untuk mengakses jenis soal A adalah 1,5 MB dan jenis soal B adalah 1,5 MB.

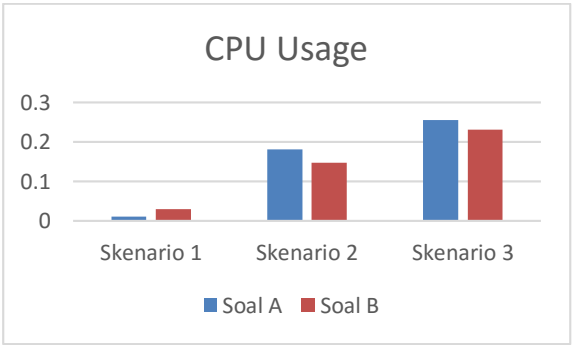


Gambar 4. Grafik Hasil Pengujian *Memory Usage*

Pada Gambar 4 merupakan grafik dari hasil pengujian *memory usage*. Tidak ada perbedaan yang signifikan antara ketiga skenario kecuali pada skenario 1 untuk jenis soal B yang membutuhkan *memory usage* sedikit lebih besar daripada lainnya.

Berdasarkan kasus uji pada tabel 4 didapatkan hasil sebagai berikut :

a. Pada skenario 1 rata-rata *CPU Usage* yang dibutuhkan untuk mengakses jenis soal A adalah 0,011 % dan jenis soal B adalah 0,0308 %.
b. Pada skenario 2 rata-rata *CPU Usage* yang dibutuhkan untuk mengakses jenis soal A adalah 0,1822 % dan jenis soal B adalah 0,1484 %.
c. Pada skenario 3 rata-rata *CPU Usage* yang dibutuhkan untuk mengakses jenis soal A adalah 0,2556 % dan jenis soal B adalah 0,2318 %.



Gambar 5. Grafik Hasil Pengujian *CPU Usage*

Pada Gambar 5 merupakan grafik hasil pengujian *CPU Usage*. Terdapat perbedaan yang cukup signifikan antara ketiga skenario. Semakin banyak pengguna yang mengakses sistem maka semakin besar pula *CPU Usage* yang dibutuhkan.

5. Kesimpulan

Berdasarkan hasil pengujian terhadap performa *server* dapat disimpulkan dari dua jenis soal yang diujikan bahwa semakin besar pengguna yang mengakses aplilasi maka semakin besar juga *CPU Usage* yang dibutuhkan yaitu paling besar adalah 0,2556 %. Untuk pengujian *page load time* dan *memory usage* tidak ditemukan perbedaan yang cukup signifikan ketika sistem digunakan oleh satu pengguna maupun banyak pengguna yaitu antara 0,208392 detik sampai 0,406842 detik untuk *page load time* dan antara 1,49 % sampai 1,56 % untuk *memory usage*.

Daftar Pustaka

Afandi, A. (2012). Implementasi Algoritma Jaro Winkler Distance Untuk Aplikasi Penilaian Otomatis.

Eko Sakti, M. A. (2016). Comparative Analysis Of String Similarity And Corpus-Based Similarity For Automatic Essay Scoring System On E-Learning Gamification.

Fuat, R. (2011). Sistem Penilaian Uraian Pendek Otomatis pada E-Learning dengan Metode Cosine Similarity.

Gomma, W. (2012). Short Answer Grading Using Similarity and Corpus Based Similarity.

Kurniawan, A. (2014). Perancangan dan Pembuatan Aplikasi Pencarian Informasi Beasiswa dengan Menggunakan Cosine Similarity.

Sakti, E. (2015). Elearning Berbasis Gamification.

Thada, V. (2013). *Comparison of Jaccard, Dice, Cosine Similarity.*

Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) p-ISSN: 2355-7699
Vol. 3, No. 4, Desember 2016, hlm. 253-258 e-ISSN: 2528-6579

PENGGUNAAN METODE *FUZZY* DALAM PENILAIAN TINGKAT KEMAMPUAN NON-AKADEMIK MAHASISWA MELALUI SATUAN KREDIT KEGIATAN MAHASISWA

Ni Putu Linda Santiari[1]

[1]STMIK STIKOM Bali
Email: [1]santiarilinda@yahoo.id

(Naskah masuk: 1 Oktober 2016, diterima untuk diterbitkan: 26 Desember 2016)

Abstrak

Pemberian point kegiatan non-akademik di STIKOM Bali dilakukan dengan menggunakan point SKKM (Satuan Kredit Kegiatan Mahasiswa).Satuan Kredit Kegiatan Mahasiswa (SKKM) adalah suatu pengakuan dan penilaian terhadap kegiatan yang diikuti oleh mahasiswa STMIK STIKOM Bali dalam pengembangan kegiatan kemahasiswaan.Pengakuan dan penilaian kegiatan yang diikuti dinyatakan dalam bentuk Satuan Kredit Kegiatan Mahasiswa (SKKM). Besarnya pembobotan SKKM yang diberikan disesuaikan dengan jenis dan pelaksanaan kegiatan yang diikuti. Saat ini penghitungan point SKKM masih dilakukan manual dan belum bisa menilai tingkat kemampuan non akademik mahasiswa. Terkait belum adanya cara untuk menilai tingkat prestasi non-akademik mahasiswa sehingga bagian kemahasiswaan tidak dapat menentukan kebijakan apa yang harus diambil untuk mengukur tingkat prestasi non akademik mahasiswa. Berdasarkan informasi di atas, penulis mencoba untuk membuat dan merancang logika fuzzy untuk membantu bagian kemahasiswaan dalam mengevaluasi tingkat prestasi non akademik mahasiswa melalui Satuan Kredit Kegiatan Mahasiswa. Penulis berharap hasil dari evaluasi ini dapat membantu bagian kemahasiswaan dalam menilai tingkat prestasi non akademik mahasiswa melalui Satuan Kredit Kegiatan Mahasiswa dan mengambil langkah berikutnya dari penilaian tersebut. Data sample yang digunakan adalah 10 mahasiswa yang memiliki SKKM.

Kata kunci: Fuzzy, SKKM, Non-akademik

Abstract

Granting point non-academic activities in Bali STIKOM done using point SKKM (Satuan Kredit Kegiatan Mahasiswa). SKKM is a recognition and assessment of the activities pursued by students STMIK STIKOM Bali in the development and assessment activities of student affairs. Recognition of the activities pursued expressed in the form of SKKM. The amount of weighting SKKM is adjusted to the type and implementation of the activities pursued. Metering point SKKM still done manually and not able to assess the level of non-academic abilities of students. Related lack way to assess the non-academic achievements of the student so that the student can not determine what policy should be taken to measure the level of non-academic achievements of students. Based on the above information, the author tries to create and design a fuzzy logic to help the student section in evaluating the level of non-academic achievements of students through the SKKM. The authors hope the results of this evaluation can help the Student Council in assessing the level of non-academic achievements of students through the Student Activity Credit Unit and take the next step of the assessment. The data sample used was 10 students who have SKKM.

Keywords: Fuzzy, SKKM, Non-academic

1. PENDAHULUAN

Logika fuzzy merupakan salah satu komponen pembentuk soft computing. Logika fuzzy pertama kali diperkenalkan oleh Prof. Lotfi A. Zadeh pada tahun 1965. Dasar dari logika fuzzy adalah teori himpunan fuzzy. Pada teori himpunan fuzzy, peranan derajat keanggotaan sebagai penentu keberadaaan elemen dalam suatu himpunan sangatlah penting. Nilai keanggotaan atau derajat keanggotaan atau *membership function* menjadi ciri utama dari penalaran dengan logika fuzzy tersebut (Kusuma Dewi, 2010).

Satuan Kredit Kegiatan Mahasiswa (SKKM) adalah suatu pengakuan dan penilaian terhadap kegiatan yang diikuti oleh mahasiswa STMIK STIKOM Bali dalam pengembangan kegiatan kemahasiswaan. Pengakuan dan penilaian kegiatan yang diikuti dinyatakan dalam bentuk Satuan Kredit Kegiatan Mahasiswa (SKKM). Besarnya pembobotan SKKM yang diberikan disesuaikan

dengan jenis dan pelaksanaan kegiatan yang diikuti (SKKM STIKOM Bali, 2015).

Penulis dalam hal ini menggunakan metode Fuzzy dibandingkan dengan metode metode yang lain karena ada beberapa kelebihan. metode fuzzy mamdani dan sesuai digunakan dalam penelitian untuk menilai tingkat kemampuan non-akademik mahasiswa melalui satuan kredit kegiatan mahasiswa diantaranya adalah pembentukan himpunan fuzzy yang sesuai dengan kebutuhan dalam penelitian, komposisi aturan aturan yang sesuai dan penegasan (defuzy) untuk mencari nilai yang bergerak secara halus sehingga perubahan dari suatu himpunan fuzzy juga akan berjalan secara halus dan lebih mudah dalam perhitungan. Penulis menggunakan metode fuzzy mamdani karena metode ini telah berhasil dipergunakan dalam penelitian Syaeful Anas Aklani tahun 2014 yang berjudul Metode Fuzzy Logic Untuk Evaluasi Kinerja Pelayanan Perawat (Studi Kasus: RSIA Siti Hawa Padang). Pada penelitian Evaluasi Kinerja Pelayanan Perawat menggunakan metode fuzzy mamdani dan hasil dari evalusi dapat menghasilkan suatu nilai yang dapat dipergunakan dalam mengukur kinerja perawat pada RSIA Siti Hawa Padang. Dari penelitian tersebut penulis mencoba menggunakan metode mamdani dalam permasalahan ini.

2. TINJAUAN PUSAKA

2.1 Fuzzy Logic

Logika fuzzy merupakan salah satu komponen pembentuk soft computing. Logika fuzzy pertama kali diperkenalkan oleh Prof. Lotfi A. Zadeh padatahun 1965. Dasar dari logika fuzzy adalah teori himpunan fuzzy. Pada teori himpunan fuzzy, peranan derajat keanggotaan sebagai penentu keberadaaan elemen dalam suatu himpunan sangatlah penting. Nilai keanggotaan atau derajat keanggotaan atau *membership function* menjadi ciri utama dari penalaran dengan logika fuzzy tersebut.
Dalam banyak hal, logika fuzzy digunakan sebagai suatu cara untuk memetakan permasalahan dari input menuju *output* yang diharapkan. Beberapa contoh yang dapat diambil antara lain (Dini Rusmiyati Andari, 2009):

1) Manajer pergudangan mengatakan kepada manajer produksi seberapa banyak persediaan barang pada akhir minggu ini, kemudian manajer produksi akanmenetqapkan jumlah barang yang akan diproduksi esok hari.

2) Seorang pegawai melakukan tugasnya dengan kinerja yang sangat baik, kemudia atas akanmemberikan reward, yang sesuai dengan kinerja pegawai tersebut.

Fuzzy Mamdani
Metode mamdani sering dikenal dengan metode Max-min. Metode ini diperkenalkan oleh Ebrahim Mamdani pada tahun 1975. Untuk mendapatkan *output*, diperlukan 4 tahapan (Fithriani Matondang dkk, 2010):

1) Pembentukan himpunan fuzzy
Pada metode mamdani baik variabel inputmaupun variabel *output* dibagi menjadi satu atau lebih himpunan fuzzy.

2) Aplikasi fungsi implikasi
Pada metode mamdani, fungsi implikasi yang digunakan adalah min.

3) Kompisisi aturan
Tidak seperti penalaran momoton, apabila sistem terdiri dari beberapa aturan, maka inferensi diperoleh dari kumpulan dan kolerasi.Ada tiga metode yang digunakan dalam melakukan inferensi sistem fuzzy, yaitu max, additive dan probabilistik OR (probor).

Metode Max (Maximum)
Pada metode ini, solusi himpunan fuzzy diperoleh dengan cara mengambil nilai maksimum atruan, kemudian menggunakannya untuk memodifikasi daerah fuzzy, dan mengaplikasikannya ke *output* dengan menggunakan operator OR (union). Jika semua proposisi telah dievaluasi, maka *output* akan berisi suatuhimpunan fuzzy yang merefleksikan komtribusi dari tiap tiap proposisi.
Secara umum dapat dituliskan:

$$\mu sf[xi] = \max (\mu usf [Xi], \mu kf [Xi]) \qquad (1)$$

Dengan :
$\mu sf[Xi]$ = nilai keanggotaannya solusi fuzzy sampai aturan ke-*i*.
$\mu kf[Xi]$ = nilai keanggotaan konsekuan fuzzy aturan ke-*i*.

Misalkan ada 3 aturan (proposisi) sebagai berikut:

[R1] IF Biaya Produksi RENDAH and Permintaan NAIK THEN Produksi Barang BERTAMBAH;

[R2] IF Biaya Produksi STANDARTHEN Produksi Barang NORMAL;

[R3] IF Biaya Produksi TINGGI and Permintaan TURUN THEN Produksi barang BERKURANG;

Proses inferensi dengan menggunakan metode Max dalam melakukan komposisi ini sering disebut dengan nama MAX-MIN atau MIN-MAX atau MAMDAI

Metode Additive (Sum)
Pada metode ini, solusi himpunan fuzzy diperoleh dengan cara melakukan bounded-sum terhadap semua *output* daerah fuzzy secara umum dituliskan:

$$\mu sf(xi)=\min(1,\mu sf(xi)+\mu kf(xi) \quad (2)$$

dengan:
μsf[*Xi*] = nilai keanggotaannya solusi fuzzy sampai aturan ke-*i*
μkf[*Xi*] = nilai keanggotaan konsekuan fuzzy aturan ke-*i*

Metode Probalistik OR (probor)
Pada metode ini, solusi himpunan fuzzy diperoleh dengan cara melakukan product terhadap semua *output* daerah fuzzy. Secara umum dituliskan:

$$\mu sf(xi)=(\mu sf(xi)+\mu kf(xi))-(\mu sf(xi)*\mu kf(xi)) \quad (3)$$

dengan:
μsf[*Xi*] = nilai keanggotaannya solusi fuzzy sampai aturan ke-*i*
μkf[*Xi*] = nilai keanggotaan konsekuan fuzzy aturan ke-*i*

Penegasan (defuzzy)
Input proses defuzzifikasi adalah suatu himpunan fuzzy yang diperoleh dari komposisi aturasn aturan fuzzy, sedangkan *output* yang dihasilkan merupakan suatu bilangan pada domain himpunan fuzzy dalam range tertentu, maka harus dapat diambil suatu nilai crisp tertentu sebagai *output*.

2.2 Satuan Kredit Kegiatan Mahasiswa (SKKM)

Tujuan dari dibuatnya Satuan Kredit Kegiatan Mahasiswa (SKKM) adalah meningkatkan peranan dan partisipasi aktif mahasiswa dalam setiap kegiatan kemahasiswaan dalam rangka peningkatan wawasan akademis dan non akademis para mahasiswa, untuk meningkatkan rasa kebersamaan dan persaudaraan antara mahasiswa, dan memberikan pernghargaan atas partisipasi aktif mahasiswa dalam setiap kegiatan kemahasiswaan.

Dalam penilaian Satuan Kredit Kegiatan Mahasiswa (SKKM) ada beberapa indikator yang dijadikan alat ukur untuk penilaian SKKM diantaranya adalah:

1) Bidang Akademik: merupakan bidang penilaian yang melibatkan kepanitian dan partisipasi mahasiswa dalam pembuatan atau penyajian makalah / moderator ilmiah, melakukan penelitian / karyatulis, seminar dan workshop.
2) Bidang Minat Bakat dan Olahraga: merupakan bidang penilaian yang melibatkan kepanitian dan partisipasi mahasiswa dalam keiukusertaan dalam turnamen / kompetisi minat dan bakat yang diselenggarakan oleh kampus maupun diluar kampus
3) Bidang Pengabdian Masyarakat : merupakan bidang penilaian yang melibatkan kepanitian dan partisipasi mahasiswa dalam keikutsertaan mahasiswa pada pengabdian masyarakat, bakti sosial, donor darah serta kegiatan-kegiatan pengabdian kepada masyarakat
4) Kegiatan Wajib : merupakan bidang penilaian yang melibatkan kepanitian dan partisipasi mahasiswa dalam kegiatan wajib yang diikuti oleh mahasiswa seperti jalan sehat, pentas musik serta gema mahasiswa teknologi informasi (GMTI)

Jadi ada beberapa faktor dalam penilaian tingkat kemampuan non-akademik mahasiswa melalui satuan kredit kegiatan mahasiswa.

3. PEMBAHASAN

3.1 Analisa Data

Dalam analisa ini penulis menganalisa dan mengelompok kelompokkan data untuk memudahkan dalam mengerjakan perancangan sistem yang telah direncanakan sebelumnya sesuai dengan variabel variabel yang dibutuhkan, guna untuk menganalisa data yang diperlukan dalam perancangan sistem ini. FIS (*Fuzzy Inference System*) dibangun dengan dua metode, yaitu mamdani dan metode sugeno, Keluaran fuzzy mamdani berupa *fuzzy set* dan bukan sekedar inverse dari fungsi keanggotaan *output*. Dengan kata lain untuk menghitung harga keluaran dari IF-THEN rule, metode mamdani harus menghitung luas dibawah kurva *fuzzy set* pada bagian keluaran (THEN-part). Selanjutnya dalam proses defuzzifikasi, metode mamdani harus menghitung rata rata (*centroid*) luas yang diboboti dari semua fuzzy set keluaran dari rule, kemudian mengisikan rata-rata tersebut ke variabel keluaran FIS.

Di dalam tahap menganalisa dan merancang sistem terdapat 5 variabel yang digunakan dan di perlukan antara lain: Kepemimpinan, administrasi, manajemen, *teamwo*rk, partisipan.

Untuk nilai kepemimpinan dibagi menjadi 4 bagian seperti pada Tabel 1. berikut:

Tabel 1. Nilai Kepemimpinan

Semesta	Himpunan Fuzzy	Domain Nilai
Point 0 - 100	Kurang	0 -50
	Cukup	40 70
	Baik	60 - 90
	Sangat Baik	80-100

Function Variabel Kepemimpinan

Terdapat 4 himpunan fuzzy untuk variabel kepemimpinan antara lain : kurang, cukup, baik, dan sangat baik. . Himpunan fuzzy tidak pernah memiliki domain (0-50) dengan derajat keanggotaan kurang tertinggi terdapat pada nilai 40, jika nilai

variabel semakin tinggi dan melebihi nilai 40 maka semakin mendekati kadang-kadang, himpunan fuzzy kurang dipresentasikan dengan bahu kiri, himpunan fuzzy kurang sebagai berikut:

$$\mu\text{kurang}[X_1] \begin{cases} 1; X_1 \le 40 \\ \frac{40-X1}{5};\ 40 \le X_1 \le 50 \\ 0; X_1 \ge 0 \end{cases}$$

Untuk himpunan fuzzy cukupmempunyai domain (40 – 70) dengan derajat keanggotan kadang kadang, tertinggi nilainya terdapat pada 55, jika nilai variabel semakin tinggi dan melebihi nilai 55 maka semakin mendekati sering. Himpunan fuzzy kadang kadang diimplementasikan dengan fungsi keanggotaan segitiga, himpunan fuzzy kadang kadang sebagai berikut;

$$\mu\text{cukup}[X_1] \begin{cases} 0; X_1 \le 40 \text{ atau } X_1 \ge 70 \\ \frac{X1-40}{15};\ 40 \le X_1 \le 55 \\ \frac{70-X1}{15}; 55 \le X_1 \le 70 \end{cases}$$

Untuk himpunan fuzzy baik mempunyai domain (60–90) dengan derajat keanggotan sering, tertinggi nilainya terdapat pada 75, jika nilai variabel semakin tinggi dan melebihi nilai 75 maka semakin mendekati selalu. Himpunan fuzzy sering di implementasikan dengan fungsi keanggotaan segitiga, himpunan fuzzy sering sebagai berikut;

$$\mu\text{baik}[X_1] \begin{cases} 0; X_1 \le 60 \text{ atau } X_1 \ge 90 \\ \frac{X1-60}{15};\ 60 \le X_1 \le 75 \\ \frac{90-X1}{15}; 75 \le X_1 \le 90 \end{cases}$$

Sedangkan untuk himpunan fuzzy sangat baik mempunyai domain (80–100) dengan derajat keanggotaan selalu, tertinggi nilainya terdapat pada 90, apabila nilai kurang dari 90 maka mendekati sering, himpunan fuzzy selalu di presentasikan dengan bahu kanan.

$$\mu\text{sangatbaik}[X_1] \begin{cases} 0; X_1 \le 80 \\ \frac{X1-60}{10};\ 70 \le X_1 \le 90 \\ 1; X_1 \ge 90 \end{cases}$$

Contoh Penerapan Fuzifikasi :

Total point secara keseluruhan pada kepemimpinan: 70, administrasi: 75, manajemen: 89, teamwork: 80, dan partisipan: 70.

1. Kepemimpinan = 70

$$\mu\text{kurang}[70] \begin{cases} 0; X_1 \le 40 \\ \frac{40-X1}{5};\ 40 \le X_1 \le 50 \\ 0; X_1 \ge 0 \end{cases}$$

$$\mu\text{cukup}[70] \begin{cases} 0; X_1 \le 40 \text{ atau } 64 \ge 70 \\ \frac{70-40}{15};\ 40 \le 64 \le 55 \\ \frac{70-70}{15}; 55 \le 64 \le 70 \\ =0 \end{cases}$$

$$\mu\text{baik}[70] \begin{cases} 0;\ X_1 \le 60 \text{ atau } X_1 \ge 90 \\ \frac{70-60}{15};\ 60 \le X_1 \le 75 \\ \frac{90-70}{15}; 75 \le X_1 \le 90 \\ =0{,}67 \end{cases}$$

$$\mu\text{sangatbaik}[70] \begin{cases} 0;\ X_1 \le 80 \\ \frac{X1-60}{10};\ 70 \le X_1 \le 90 \\ 1;\ X_1 \ge 90 \end{cases}$$

Administrasi: 75

$$\mu\text{kurang}[75] \begin{cases} 1;\ 75 \le 30 \\ \frac{30-75}{5};\ 30 \le X_3 \le 40 \\ 0; X_3 \ge 0 \end{cases}$$

$$\mu\text{cukup}[75] \begin{cases} 0; X_3 \le 30 \text{ atau } X_3 \ge 60 \\ \frac{X3-30}{15};\ 30 \le X_3 \le 45 \\ \frac{60-75}{15}; 45 \le X_3 \le 60 \\ =0 \end{cases}$$

$$\mu\text{baik}[75] \begin{cases} 0;\ 75 \le 50 \text{ atau } X_3 \ge 90 \\ \frac{75-50}{20};\ 50 \le X_3 \le 70 \\ \frac{90-75}{20}; 70 \le X_3 \le 90 \\ =0{,}75 \end{cases}$$

$$\mu\text{sangatbaik}[75] \begin{cases} 0;\ X_3 \le 70 \\ \frac{X3-70}{15};\ 70 \le X_3 \le 85 \\ 1;\ X_3 \ge 85 \end{cases}$$

Manajemen = 89

$$\mu\text{kurang}[89] \begin{cases} 1;\ X_2 \le 40 \\ \frac{40-89}{10};\ 40 \le 89 \le 50 \\ 0; X_2 \ge 0 \end{cases}$$

$$\mu cukup[89] \begin{cases} 0; X_2 \le 40 \text{ atau } X_2 \ge 70 \\ \frac{X3-40}{15}; 40 \le X_2 \le 55 \\ \frac{70-X2}{15}; 55 \le X_2 \le 70 \end{cases}$$

$$\mu baik[89] \begin{cases} 0; X_2 \le 60 \text{ atau } X_2 \ge 90 \\ \frac{89-60}{15}; 60 \le X_2 \le 75 \\ \frac{90-89}{15}; 75 \le X_2 \le 90 \\ =0{,}06 \end{cases}$$

$$\mu sangatbaik[89] \begin{cases} 0; 89 \le 80 \\ \frac{89-80}{10}; 70 \le 89 \le 90 \\ 1; 89 \ge 90 \end{cases}$$

Teamwork = 80

$$\mu kurang\ [80] \begin{cases} 1; 80 \le 30 \\ \frac{30-X4}{5}; 30 \le X_4 \le 40 \\ 0; X_4 \ge 0 \end{cases}$$

$$\mu cukup[80] \begin{cases} 0; 80 \le 30 \text{ atau } X_4 \ge 60 \\ \frac{80-30}{15}; 30 \le X_4 \le 45 \\ \frac{60-30}{15}; 45 \le X_4 \le 60 \end{cases}$$

$$\mu baik[80] \begin{cases} 0; X_4 \le 50 \text{ atau } X_4 \ge 90 \\ \frac{X4-50}{20}; 50 \le X_4 \le 70 \\ \frac{90-80}{20}; 70 \le X_4 \le 90 \end{cases}$$

$$\mu sangatbaik[80] \begin{cases} 0; 80 \le 70 \\ \frac{80-70}{15}; 70 \le 80 \le 85 \\ 1; X_4 \ge 85 \end{cases}$$

Partisipan = 70

$$\mu kurang\ [70] \begin{cases} 0; X_1 \le 40 \\ \frac{40-X1}{5}; 40 \le X_1 \le 50 \\ 0; X_1 \ge 0 \end{cases}$$

$$\mu cukup[70] \begin{cases} 0; X_1 \le 40 \text{ atau } 64 \ge 70 \\ \frac{70-40}{15}; 40 \le 64 \le 55 \\ \frac{70-70}{15}; 55 \le 64 \le 70 \\ =0 \end{cases}$$

$$\mu baik[70] \begin{cases} 0; X_1 \le 60 \text{ atau } X_1 \ge 90 \\ \frac{70-60}{15}; 60 \le X_1 \le 75 \\ \frac{90-70}{15}; 75 \le X_1 \le 90 \\ =0{,}67 \end{cases}$$

$$\mu sangatbaik[70] \begin{cases} 0; X_1 \le 80 \\ \frac{X1-60}{10}; 70 \le X_1 \le 90 \\ 1; X_1 \ge 90 \end{cases}$$

3.2 Penalaran (Inferensi)

Tahap ini merupakan penentuan *rule-rule* dari sistem logika *fuzzy*, aturan-aturan dapatdibentuk untuk menyatakan relasi antara input danoutput. Tiap aturan merupakan implementasi.Operator yang digunakan untuk menghubungkanaturan-aturan *input* adalah operator *And* yang menggambarkan antara input-output adalah IF –THEN.

If (Kepemimpinan is Baik) and (Administrasi is SangatBaik) and (Manajemen is Baik) and (Teamwork is Baik) and (Partisipan is Baik) then (output1 is Aktif)

α- hasil point skkm = μKPbaik ∩ μADsangatbaik ∩ μMMbaik∩μTWbaik∩μPSbaik
=μKPbaik(70)∩μADsangatbaik (75) ∩ μMMbaik (89) ∩ μTWbaik(80) ∩ μPSbaik (70)
= min (0,67 ; 0,75 ; 0,06 ; 0,50 ; 0,67)
= 0,06

3.3 Defuzifikasi

Tahap ini disebut juga tahap penegasaninput dan proses. Penegasan ini adalah suatu himpunan kabur yang diperoleh dari komposisiaturan-aturan kabur, sedangkan output yang dihasilkan merupakan suatu bilangan pada domain himpunan kabur tersebut.

Untuk predikat mahasiswa dapat dilihat dari Tabel 2. berikut:

Tabel 2. Tabel Predikat

Point	Predikat	Domain Nilai
Point 0 - 100	Kurang Aktif	0 -50
	Cukup Aktif	40 70
	Aktif	60 - 90
	Sangat Aktif	80-100

Hasil Rule seperti data di atas adalah;

$$Z = \frac{70.0{,}67+75.0{,}75+89.0.06+80.0{,}5+70.0{,}67}{0.67+0.75+0.06+0{,}5+0{,}67}$$

$$Z = \frac{46.9+56.25+5.34+40+46{,}9}{2.65}$$

$$Z = \frac{195{,}39}{2.65}$$

$$Z = 73.73$$

Maka dari percobaan diatas terdapat hasil nilai 73.73 yang termasuk dalam predikat mahasiswa yang aktif terlihat dari domain antara 60-90 tergolong aktif.

4. KESIMPULAN

Dari hasil penelitian tersebut maka metode fuzzy untuk penelitian penilaian tingkat kemampuan non-akademik mahasiswa melalui satuan kredit kegiatan mahasiswa, sesuai hasil yang di inginkan dan metode fuzzy dapat mengevaluasi mahasiswa dan membantu dalam penilaian tingkat kemampuan non-akademik mahasiswa dilihat dari hasil percobaan dengan total point secara keseluruhan pada kepemimpinan: 70, administrasi: 75, manajemen: 89, teamwork: 80, dan partisipan: 70 diperoleh hasil evaluasi mahasiswa dengan predikat aktif.

5. DAFTAR PUSTAKA

AKLANI, SYAEFUL ANAS. 2014. Metode Fuzzy Logic Untuk Evaluasi Kinerja Pelayanan Perawat (Studi Kasus: RSIA Siti Hawa Padang). Pendidikan Informatika STKIP PGRI Sumbar.

AMRI, F., NABABAN, E.B. & SYAHPUTRA, M.F. 2012. *Artificial Bee Colony Algorithm untuk menyelesaikan Travelling Salesman Problem*. Jurnal Dunia Teknologi Informasi. 1(1): 8-13.

ANDARI, RR. DINI RUSMIYATI. 2009. Aplikasi Fuzzy Database Evaluasi Kinerja Pegawai di SMK Negeri 02 Bangkalan Menggunakan JSP.

ARIFIN, ZAENAL. 2009. Evaluasi Pembelajaran. Bandung: Remaja Rosdakarya hal 2.

KEMAHASISWAAN. 2015. Buku Pedoman Satuan Kredit Kegiatan Mahasiswa. Denpasar.

KUSUMA DEWI. 2010. Aplikasi Logika Fuzzy Untuk Pendukung Keputusan Edisi 2, Graha Ilmu.

MATONDANG, FITHRIANI dkk. 2010. Fuzzy Logic Metode Mamdani Untuk Membantu Diagnosa Dini Autism Spectrum Disorder. UIN Maulana Malik Ibrahim Malang

PURWANTO, M. NGALIM. 2010. Prinsip-prinsip dan Teknik Evaluasi Pengajaran. Bandung: Remaja Rosdakarya hal 3.

RAKHMAWATI, ESA. 2013. *Pengembangan Penilaian Kinerja Siswa (Students Performance Assessment) Dalam Menemukan Rumus Pythagoras.* Undergraduate thesis, UIN Sunan Ampel Surabaya.

Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) p-ISSN: 2355-7699
Vol. 3, No. 4, Desember 2016, hlm. 259-264 e-ISSN: 2528-6579

OPTIMASI PENJADWALAN PENGERJAAN *SOFTWARE* PADA *SOFTWARE HOUSE* DENGAN *FLOW-SHOP PROBLEM* MENGGUNAKAN *ARTIFICIAL BEE COLONY*

Muhammad Fhadli[1], Daneswara Jauhari[2], Dhimas Anjar Prabowo[3], Anang Hanafi[4], Aryeswara Sunaryo[5], Imam Cholissodin[6]

[1,2,3,4,5,6]Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]muhammadfhadli20@gmail.com, [2]daneswarajauhari@gmail.com, [3]dhimasanjar@gmail.com, [4]ananghanafi13@gmail.com, [5]ary3swara@gmail.com, [6]imamcs@ub.ac.id

(Naskah masuk: 1 Oktober 2016, diterima untuk diterbitkan: 26 Desember 2016)

Abstrak

Penelitian ini mengusulkan sebuah implementasi terkait optimasi penjadwalan pengerjaan *software* pada *software house* dengan *Flow-Shop Problem* (FSP) menggunakan algoritma *Artificial Bee Colony* (ABC). Dimana dalam FSP dibutuhkan suatu solusi untuk menyelesaikan suatu *job*/*task* dengan meminimalkan total *cost* yang dikeluarkan. Terdapat *constraint* yang perlu diperhatikan dalam objek permasalahan penelitian ini, yaitu lama waktu penyelesaian keseluruhan projek *software* yang tidak pasti. Dalam penelitian ini akan disusun sebuah representasi solusi yaitu berupa urutan pengerjaan projek dengan total waktu pengerjaan yang minimum. Pengujian akan dilakukan dengan tiga kali percobaan untuk setiap kondisi uji coba, yaitu uji coba batas parameter iterasi dan uji coba batas parameter limit. Dari hasil pengujian didapatkan bahwa penggunaan algoritma yang dibahas dalam penelitian ini bisa mengurangi waktu pengerjaan jika jumlah iterasi dan jumlah *colony* diperbesar.

Kata kunci: *optimasi, flow-shop problem, artificial bee colony, swarm intelligence, meta-heuristik.*

Abstract

This research proposed an implementation related to software execution scheduling process at a software house with Flow-Shop Problem (FSP) using Artificial Bee Colony (ABC) algorithm. Which in FSP required a solution to complete some job/task along with its overall cost at a minimum. There is a constraint that should be kept to note in this research, that is the uncertainty completion time of its jobs. In this research, we will present a solution that is a sequence order of project execution with its overall completion time at a minimum. An experiment will be performed with 3 attempts on each experiment conditions, that is an experiment of iteration parameter and experiment of limit parameter. From this experiment, we concluded that the use of this algorithm explained in this paper can reduce project execution time if we increase the value of total iteration and total colony.

Keywords: *optimization, flow-shop problem, artificial bee colony, swarm intelligence, meta-heuristic.*

1. PENDAHULUAN

Scheduling atau penjadwalan merupakan pekerjaan untuk mengalokasikan sumber daya yang terbatas dari suatu pekerjaan untuk mengefisiensikan waktu pengerjaan (Garrido dkk., 1994). *Job Shop Problem* merupakan suatu permasalahan dimana dibutuhkan suatu solusi untuk menyelesaikan suatu *job*/*task* dengan meminimalkan total *cost* yang dikeluarkan. Pada penelitian ini, terdapat *constraint* yang perlu diperhatikan yaitu ketidakpastian waktu proses penyelesaian suatu *job*.

Ketidakpastian waktu proses penyelesaian tentunya berhubungan erat dengan bidang pekerjaan tertentu seperti *software house*, dsb. Untuk *constraint* pertama berupa masalah ketidakpastian dalam waktu pemrosesan, dapat diselesaikan menggunakan algorima kombinatorial *Artificial Bee Colony*.

Penulis menemukan salah satu penelitian yang berhubungan dengan konsep dasar dari penjadwalan. Penelitian ini mendiskusikan permasalahan *Constraint Satisfaction Problem* (CSP) yang didasari oleh konsep *slack* untuk nilai dan variabel tertentu yang harus dilakukan oleh suatu operasi (Garrido dkk., 1994). Penelitian ini menyimpulkan bahwa metode heuristik dapat menyelesaikan beberapa jenis permasalahan yang sulit diselesaikan dengan menggunakan CSP tradisional.

Penelitian sebelumnya yang terkait dengan penjadwalan juga pernah dilakukan oleh Bruker dan Schile (Gao dkk., 2016) yang merupakan penggagas dalam penelitian tentang *Flexible Job Shop Problem* (FJSP) dengan mengajukan penelitian berupa algoritma *polynomial* untuk 2 *jobs* dan penggunaan mesin identik pada sebuah FJSP.

Penelitian lain dengan permasalahan FJSP dengan *uncertaity processing time* dan *new job insertion* pernah dilakukan oleh Gao, dkk. (2016) dengan menggunakan *Artificial Bee Colony* (ABC). Penelitian ini menggambarkan dan membandingkan

beberapa metode solusi terbaru dan membahas strategi pengembangan. Tujuannya adalah untuk meminimalkan nilai dari *fuzzy completion time.* Pada penelitian yang menjadi rujukan penulis, penelitian tersebut menggunakan *Two-stage Artificial Bee Colony* (TABC) dengan beberapa pengembangan dan hasilnya menunjukkan bahwa metode tersebut mampu bersaing dengan beberapa metode yang sudah ada sebelumnya.

Dari latar belakang tersebut, kami mencoba menerapkan prinsip-prinsip tersebut dengan merubah domain permasalahan menjadi penjadwalan pengerjaan *software.* Penelitan ini penulis beri judul Optimasi Penjadwalan Pengerjaan *Software* pada *Software House* dengan *Flow-Shop Problem* Menggunakan *Artificial Bee Colony*. Penelitian ini menggunakan data pegerjaakn *project* yang statis sehingga lebih efisien jika tidak menggunakan *two-stage artificial bee colony*, diharapkan penelitian yang penulis lakukan ini bisa mejadi rujukan pada penelitian-penelitian lain dengan topik yang berkaitan.

2. DASAR TEORI

2.1 Data Penelitian Projek

Penelitian ini menggunakan *dataset* yang didapat dari perusahaan *software house* GUMCODE, Malang, Jawa Timur. Data yang penulis peroleh berupa 10 daftar projek berserta *timeline* pengerjaan setiap projek. Data ini penulis konversikan kedalam satuan tabel dan bisa dilihat pada Tabel 2.1 berikut. Tabel ini menunjukkan lama pengerjaan setiap tahap pada suatu projek dalam satuan minggu.

Tabel 2.1 *Dataset* pengerjaan projek GUMCODE

Projek ke-*i*	Pengerjaan			
	Tahap 1	Tahap 2	Tahap 3	Tahap 4
1	2	2	1	2
2	1	3	3	4
3	6	9	2	3
4	1	4	3	2
5	4	4	1	2
6	3	1	0	0
7	1	2	7	3
8	1	3	4	4
9	1	2	8	1
10	1	1	5	2

2.2 Estimasi Pengerjaan *Software*

Estimasi Pengerjaan *Software* merupakan topik yang sulit. Steve McConnell menyebutnya *Black Art*, sehingga dia mengarang buku yang sangat bagus tentang topik ini, judulnya *Software Estimation, Demystifying the Black Art*. Menurut Steve, dalam membuat estimasi, ada 3 metode yang dilakukan, yaitu *count, compute, judge* (McConnell, 2006).

Karakteristik perangkat lunak tentunya memberikan kendala yang tidak sedikit terhadap estimasi yang akan dilakukan. Kerumitan dari perangkat lunak tersebut dan hal lain yang tidak kasat mata, serta sumber daya manusia pengembang perangkat lunak tidak bisa diperkirakan secara mutlak secara mekanistik seperti mesin.

2.3 *Flow-Shop Problem*

Dalam *flow-shop scheduling problem*, terdapat sejumlah *n job* dan *m* mesin, dimana setiap *job* harus diproses pada setiap mesin mulai dari mesin 1 sampai mesin *m* secara berurutan. Dengan kondisi seperti ini, tentunya akan muncul *buffer* dengan kapasitas yang tidak terbatas diantara mesin (Khorasanian, 2017). Pada permasalahan FSP ini, dapat diartikan dengan bagaimana membuat urutan pekerjaan, sehingga waktu yang dibutuhkan untuk menyelesaikan semua pekerjaan dapat diminimalkan.

2.4 Algoritma *Artificial Bee Colony*

2.4.1 Inisialisasi Parameter dan Populasi

Parameter yang harus diinisialisasi yaitu:

- Data Kasus : merupakan data pekerjaan apa saja yang harus dilakukan.
- *Colony Size* : jumlah *Employee Bee* ditambah dengan *Onlooker Bee* (Cholissodin, 2016).
- Maksimum Iterasi : merupakan banyaknya iterasi yang dilakukan.
- *Limit* : batas jumlah populasi yang kualitasnya tidak meningkat pada suatu iterasi.
- *Number of sequence* (NSE) : merupakan batasan banyak *sequence* dalam *neighborhood operator*.

Untuk melakukan inisialisasi populasi dapat dengan cara merandom urutan *project* yang harus dikerjakan, kemudian menghitung nilai *makespan* dan nilai *fitness*. Nilai *makespan* dapat dihitung dengan menggambarkan *gantt-chart*.

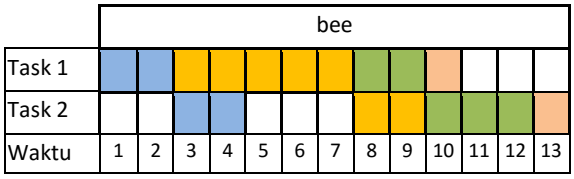
2.4.2 *Gantt Chart*

Perhitungan nilai *makespan* dilakukan menggunakan *gantt-chart.* Berikut ini penulis contohkan perhitungan *makespan* menggunakan *gantt-chart* dengan projek dan *task* yang ditunjukkan pada Tabel 2.2 dibawah ini, dengan urutan pengerjaan 2, 3, 1, 4:

Tabel 2.2 Projek dan *task*

Project	Task	
	1	2
1	2	3
2	2	2
3	5	2
4	1	1

Tabel di atas bisa kita gambarkan seperti *gantt-chart* pada Gambar 2.1 yang menunjukkan *makespan* dari *bee* tersebut adalah 13.



Gamber 2.1 *Gantt chart*

Berdasarkan contoh kasus, *project* yang pertama kali dikerjakan adalah *project* nomor 2 dimana pada *project* ini *task* 1 dan *task* 2 diselesaikan dalam 2 pekan dan *task* yang berbeda dari *project* yang sama tidak boleh dikerjakan dalam pekan yang sama sehingga pada kolom pekan pertama sampai pekan keempat di*block* dengan warna biru muda.

Hal yang sama dilakukan juga untuk *project* 3. Pada *task* 1, *project* 3 membutuhkan waktu 5 pekan untuk pengerjaannya. Oleh karena itu, *task* 2 dari *project* 3 baru dapat dijalankan pada pekan ke-8. Begitu seterusnya sampai *project* yang terakhir.

2.4.3 Rumus *Fitness*

Rumus *fitness* merupakan suatu rumus yang digunakan sebagai tolak ukur baik atau tidaknya urutan pengerjaan projek yang sedang diteliti, rumus *fitness* dapat dilihat pada persamaan 2.1 sebagai berikut:

$$fitness = {}^{1}/_{C} \quad (2.\ 1)$$

C = *cost* atau dimana dalam penelitian ini *cost* yang digunakan adalah total *makespan* dari *gantt-chart*.

2.4.4 Fase *Employeed Bee*

Pada fase ini dilakukan *update* populasi untuk setiap *Employeed Bee* dengan menggunakan *neighborhood operator*, yang terdiri dari *swap operator* (SO) dan *swap sequence* (SS). Kemudian hitung *fitness* nya, jika solusi yang baru lebih baik dari pada yang lama maka gantikan solusi lama dengan solusi baru, jika tidak tambahkan nilai trial dengan 1. Langkah-langkahnya sebagai berikut:

Swap Operator dapat dihitung dengan cara melakukan pergantian posisi antara 2 operasi. Solusi baru dapat dihitung dengan rumus (Cholissodin, 2016):

$$x_i = x_i + SO \quad (2.\ 2)$$

Swap Squences dapat dihitung dengan cara melakukan perhitungan SO sejumlah *NSE*, perhitungan SS dilakukan pada semua *Employeed Bee*. Solusi didapatkan dengan mencari nilai *fitness* terbaik pada setiap perhitungan SS, solusi baru dapat dihitung dengan rumus:

$$x_i = x_i + SS \quad (2.\ 3)$$

$$x_i = x_i + (SO_1, SO_2, SO_3, \dots, SO_n)$$

n = banyaknya SS, kemudian hitung nilai probabilitas setiap *Employeed Bee* dengan rumus (Cholissodin, 2016):

$$Prob_i = \frac{fitness(x_i)}{\sum_{k=1}^{S} fitness(x_k)} \quad (2.\ 4)$$

2.4.5 Fase *Onlooker Bee*

Untuk setiap *Onlooker Bee* didapatkan dengan cara memilih solusi yang ada pada *Employeed Bee*, dengan teknik seleksi *roulette wheel*.

Untuk menentukan solusi yang baru, dapat dilakukan dengan menggunakan *neighborhood operator*, yang terdiri dari *insert operator* (IO) dan *insert sequence* (IS). Kemudian hitung *fitness*-nya, jika solusi yang baru lebih baik dari pada yang lama maka gantikan solusi lama dengan solusi baru, jika tidak tambahkan nilai *trial* dengan 1. Langkah-langkahnya sebagai berikut:

Insert Operator dapat dihitung dengan cara melakukan *insert*, misalnya IO(1,3) artinya melakukan *insert* operasi ke 3 kedepan operasi 1. Solusi baru dapat dihitung dengan rumus (Cholissodin, 2016):

$$xi = xi + IO \quad (2.\ 5)$$

Insert Squences dapat dihitung dengan cara melakukan perhitungan IO sejumlah NSE, perhitungan IS dilakukan pada semua *Onlooker Bee*. Solusi didapatkan dengan mencari nilai *fitness* terbaik pada setiap perhitungan IS, Solusi baru dapat dihitung dengan rumus:

$$x_i = x_i + IS \quad (2.\ 6)$$

$$x_i = x_i + (IO_1, IO_2, IO_3, \dots, IO_n)$$

n = banyaknya IS

2.4.6 Fase *Scout Bee*

Setelah melalui dua fase sebelumnya, yaitu *Employeed Bee* dan *Onlooker Bee*, maka langkah selanjutnya yaitu perhitungan kualitas dari masing-masing *Employeed Bee*. Jumlah *Scout Bee* dalam fase

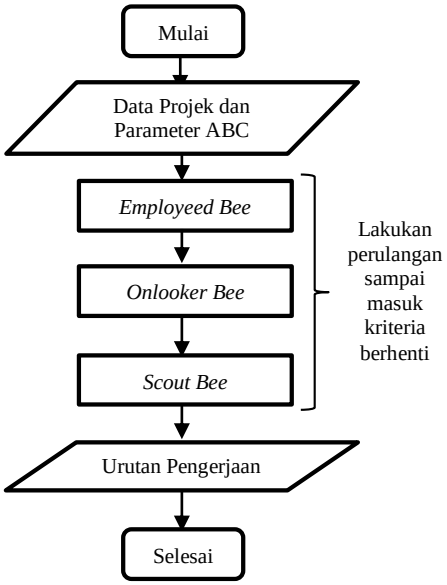
ini bersifat dinamis, tergantung pada jumlah *employeed* yang telah melebihi *limit*.

Apabila *limit* dari *Bee* yang melakukan *Improvement Solution* melebihi maksimum *limit* yang ditetapkan, maka solusi pada *Bee* tersebut dihilangkan dan digantikan dengan solusi baru yang dibuat secara *random*, kemudian memperbarui *fitness* yang dihasilkan, dan menyetel ulang *limit* menjadi 0.

3. PERANCANGAN DAN IMPLEMENTASI

3.1 Perancangan Alur Proses Algoritma

Perancangan alur proses algoritma ini kami adaptasi dari penjelasan yang telah dijelaskan pada bagian dasar teori, dimana rancangannya dapat dilihat pada Gambar 3.1.



Gambar 3.1 Diagram Alur Proses Optimasi Penjadwalan

Dimana representasi solusinya adalah indeks urutan pengerjaan projek, agar lebih jelasnya dapat dilihat contoh representasi solusi individu pada Gambar 3.2.

	<----------*representasi solusi*---------->			
P1 =	[Projek 1,	Projek 2,	…,	Projek *i*]

Gambar 3.2 Contoh representasi solusi individu

3.2 Perancangan Uji Coba

Perancangan uji coba ini dilakukan untuk mengevaluasi dan menganalisis hasil yang dihasilkan, dimana akan dilakukan 2 proses uji coba yaitu sebagai berikut:

3.2.1 Uji Coba Parameter Iterasi

Pada uji coba ini, penulis mencari nilai *fitness* yang optimal dengan melakukan perubahan sebanyak 5 kali terhadap jumlah iterasi. Percobaan ini dilakukan untuk melihat pengaruh jumlah iterasi terhadap peningkatan nilai *fitness.*

3.2.2 Uji Coba Parameter Limit

Pada uji coba ini, penulis mencari nilai *fitness* yang optimal dengan melakukan perubahan sebanyak 5 kali terhadap nilai limit. Percobaan ini dilakukan untuk melihat pengaruh nilai *limit* terhadap peningkatan nilai *fitness.*

3.3 Implementasi

Implementasi pada penelitian ini dilakukan dengan mengikuti dasar teori serta perancangan yang telah dijelaskan pada poin 2 dan 3.

Tabel 3.3 Prosedur pengerjaan FSP-ABC

Tahap 1: Membuat inisialisasi *bee* untuk tahap *employeed bee* secara acak sebanyak populasi

Tahap 2: Lakukan evaluasi *fitness* pada setiap *bee* dalam populasi

Tahap 3: Lakukan *improvement* dengan menggunakan *swap operator* (SO) pada *bee* inisialisasi, kemudian evaluasi *fitness*-nya kembali

Tahap 4: Lakukan *improvement* dengan menggunakan *swap sequence* (SS) pada *bee* hasil SO, kemudian evaluasi *fitness*-nya kembali

Tahap 5: Lakukan *selection* dengan menggunakan *roulette wheel selection* (RWS) pada *bee* hasil SS hingga terbentuk *bee* baru hasil seleksi sebanyak populasi sebagai *bee* untuk tahap *onlooker bee*

Tahap 6: Lakukan *improvement* dengan menggunakan *insertion operator* (IO) pada *bee* hasil RWS, kemudian evaluasi *fitness*-nya kembali

Tahap 7: Lakukan *improvement* dengan menggunakan *insertion sequence* (IS) pada *bee* hasil IO, kemudian evaluasi *fitness*-nya kembali

Tahap 8: Ambil individu dengan nilai *fitness* terbaik pada *bee* hasil IS sebagai *global best*/solusi terbaik

Tahap 9: Bandingkan nilai *fitness* antara *bee* inisialisasi dengan *bee* hasil IS, dengan kondisi:

- Jika maksimum nilai *trial bee* melebihi *limit* dan *bee* tidak mengalami perbaikan maka *reset* nilai *trial* menjadi 0, dan acak kembali urutan pengerjaan projeknya secara acak seperti pada tahap inisialisasi
- Jika maksimum nilai *trial bee* melebihi *limit* dan *bee* mengalami perbaikan maka *reset* nilai *trial* menjadi 0, dan biarkan urutan pengerjaan projek pada *bee* tersebut seperti pada hasil IS
- Jika maksimum nilai *trial bee* belum melebihi *limit* maka nilai *trial* tidak perlu di-*reset* dan biarkan urutan pengerjaan

projek pada *bee* tersebut seperti pada hasil IS

Kriteria berhenti dari sistem ini akan dilakukan sebanyak maksimum iterasi. Iterasi akan dilakukan sampai kriteria berhenti terpenuhi, dan selama belum terpenuhi, maka akan mengulang langkah Tahap 3.

4. PENGUJIAN DAN ANALISIS

4.1 Pengujian Parameter Iterasi

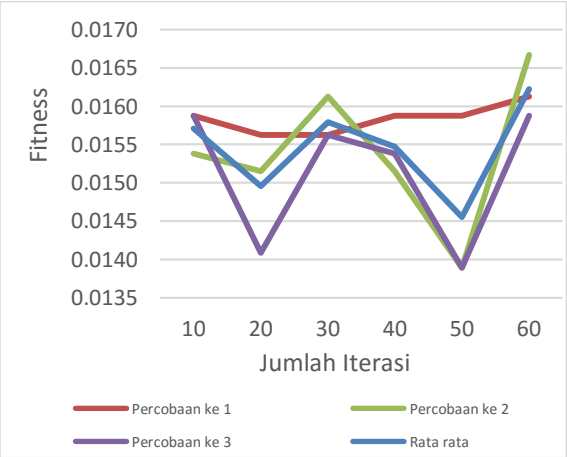
Pengujian parameter iterasi ini menggunakan jumlah *colony* sebesar 3 dan nilai *limit* diambil dari nilai terkecil pada pengujian parameter *limit*, yaitu 5. Berdasarkan hasil pengujian pada Tabel 4.1, nilai rata-rata *fitness* terkecil didapat pada iterasi 50.

Sementara itu, rata-rata nilai *fitness* terbesar bisa ditemukan pada saat iterasi 60. Sehingga jumlah iterasi bisa ditetapkan 60 agar mendapatkan nilai *fitness* maksimal.

Tabel 4.1 Pengujian Parameter Iterasi

Uji Coba Batas Parameter Iterasi				
Iterasi	**Nilai Fitness Percobaan ke-*i***			**Rata - Rata *Fitness***
	1	**2**	**3**	
10	0.0159	0.0154	**0.0159**	0.0157
20	0.0156	0.0152	0.0141	0.0150
30	0.0156	0.0161	0.0156	0.0158
40	0.0159	0.0152	0.0154	0.0155
50	0.0159	0.0139	0.0139	0.0146
60	**0.0161**	**0.0167**	**0.0159**	**0.0162**

Berikut ini adalah grafik yang menggambarkan tabel di atas.



Gambar 4.1 Pengujian Parameter Iterasi

4.2 Pengujian Parameter Limit

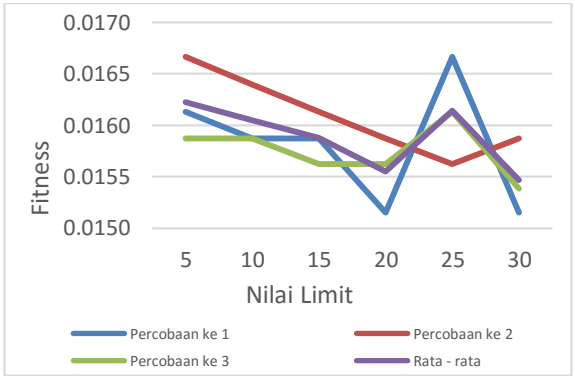
Pengujian parameter *limit* ini menggunakan jumlah *colony* sebesar 3 dan nilai iterasi diambil dari iterasi dengan *fitness* terbesar ada pengujian paramter iterasi, yaitu 60. Berdasarkan hasil pengujian pada Tabel 4.2, nilai rata-rata *fitness* terkecil didapat pada percobaan dengan limit 30.

Sementara itu, rata-rata nilai *fitness* terbesar bisa ditemukan pada saat nilai *limit* 5. Sehingga nilai iterasi bisa ditetapkan 5 agar mendapatkan nilai *fitness* maksimal.

Tabel 4.2 Pengujian Parameter *Limit*

Uji Coba Batas Parameter limit				
limit	**Nilai Fitness Percobaan ke-*i***			**Rata - Rata *Fitness***
	1	**2**	**3**	
5	0.0161	**0.0167**	0.0159	**0.0162**
10	0.0159	0.0164	0.0159	0.0160
15	0.0159	0.0161	0.0156	0.0159
20	0.0152	0.0159	0.0156	0.0155
25	**0.0167**	0.0156	**0.0161**	0.0161
30	0.0152	0.0159	0.0154	0.0155

Berikut ini adalah diagram yang menggambarkan tabel di atas.



Gambar 4.2 Pengujian Parameter *Limit*

Oleh karena itu, untuk mendapatkan nilai *fitness* yang maksimal, kita bisa mengecilkan nilai *limit* dan memperbesar nilai iterasi.

5. KESIMPULAN DAN SARAN

Artificial bee colony dengan *flow shop problem* bisa digunakan untuk menyelesaikan permasalahan penjadwalan pengerjaan *project* pada *software house* GUMCODE. Penjadwalan tersebut diuji dengan asumsi terdapat 4 mesin atau 4 tahap dalam pengerjaan setiap projek.

Pengujian ini menyimpulkan bahwa parameter yang optimal agar dapat menghasilkan nilai *fitness* terbaik adalah parameter dengan nilai iterasi 60 dan

nilai *limit* 5. Dengan kata lain, semakin besar nilai iterasi dan semakin kecil nilai *limit*, maka nilai *fitness* akan semakin baik.

Penelitian ini dapat dikembangkan dengan *job shop scheduling* yang lain ataupun bisa ditambahkan dengan pengunaan *fuzzy* seperti penelitian yang dilakukan oleh (Gao dkk., 2016).

6. DAFTAR PUSTAKA

CHOLISSODIN, I. (2016). Modul Swarm Intelligence – Semester Ganjil 2016-2017.

GAO, KAI ZHOU. dkk., 2016. Artificial bee colony algorithm for scheduling and rescheduling fuzzy flexible job shop problem with new job insertion. *Knowledge-Based Systems 109 (2016) 1-16.*

KARTHIKEYAN, A., MANIKANDAN, K. & SOMASUNDARAM, P., 2016. Economic Dispatch of Microgrid with Smart Energy Storage Systems using Particle Swarm Optimization. 2016 *International Conference on Computation of Power, Energy Information and Communication (ICCPEIC)*.

KHORASANIAN, D. & MOSLEHI, G., 2017. Two-machine flow shop scheduling problem with blocking, multi-task flexibility of the first machine, and preemption. *Computers and Operation Research*, 79(August 2016), pp.94–108. Available at: http://dx.doi.org/10.1016/j.cor.2016.09.023.

MCCONNELL, S., *Software Estimation*, Demystifying the Black Art. Washington, 2006.

Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) p-ISSN: 2355-7699
Vol. 3, No. 4, Desember 2016, hlm. 265-272 e-ISSN: 2528-6579

OPTIMASI PENJADWALAN PRAKTIKUM MENGGUNAKAN *MODIFIED REAL CODE PARTICLE SWARM OPTIMIZATION* (STUDI KASUS FAKULTAS IMU KOMPUTER UNIVERSITAS BRAWIJAYA)

Brigitta Ayu Kusuma Wardhany[1], Istiana Rachmi[2], Nur Firra Hasjidla[3], Zulianur Khaqiqiyah[4], Idham Triatmaja[5], Imam Cholissodin[6]

[1,2,3,4,5,6]Fakultas Ilmu Komputer Universitas Brawijaya
Email: [1]gittawardhany4@gmail.com, [2]istianarahmi@gmail.com, [3]firrapirraa@gmail.com, [4]zulianurhaqq@gmail.com, [5]mahdi3atmaja@gmail.com, [6]imamcs@ub.ac.id

(Naskah masuk: 1 Oktober 2016, diterima untuk diterbitkan: 26 Desember 2016)

Abstrak

Penjadwalan adalah salah satu proses dalam manajemen waktu yang di atur sedemikian rupa agar kegiatan dapat berjalan dengan lancar. Banyak algoritma yang dapat digunakan untuk menyelesaikan permasalahan penjadwalan. Pada kasus ini penulis menggunakan algoritma *Modified Real Code PSO* (M-RCPSO). Data yang digunakan terdiri dari data dosen, asisten, mahasiswa, ruangan dan waktu praktikum. Dari hasil pengujian *popsize*, pengujian iterasi, pengujian parameter kognitif dan sosial, dan pengujian parameter terbaik, didapatkan nilai rata-rata *fitness* adalah 1. Hal ini menunjukkan bahwa solusi yang didapatkan sudah mendekati optimum.

Kata kunci: *modified real code particle swarm optimization, penjadwalan*

Abstract

Scheduling is one of the time management process that well regulated so that the activities can run fluently. Many algorithms can be used to solve scheduling problems. In this case, the author uses a Modified Real Code PSO (M-RCPSO) algorithm. The data used consisted of lecturer, assistant, student, room and lab time. From the results of popsize testing, iterative testing, cognitive parameter testing, and the best parameters testing obtained the average fitness value is 1. This matter shows that the solution obtained is already approaching optimal.

Keywords: *modified real code particle swarm optimization, scheduling*

1. PENDAHULUAN

Penjadwalan adalah suatu proses yang berkaitan dengan pengaturan waktu. Pengaturan waktu sendiri memiliki arti dimana suatu kegiatan diatur sedemikian mungkin agar dapat berlangsung secara lancar dan tidak terjadi bentrok dengan kegiatan lainnya. Untuk itu diperlukan susunan jadwal kegiatan agar dapat terkoordinir dengan baik. Namun kebanyakan orang masih mengabaikan penyusunan jadwal ini dikarenakan prosesnya yang sangat membingungkan dan memakan waktu lama. Di Indonesia sendiri salah satu permasalahan tentang penjadwalan yang paling sering dihadapi adalah penjadwalan matakuliah yang didalamnya masih dibagi lagi menjadi penjadwalan praktikum dan penjadwalan ujian. Pengaturan waktu terhadap suatu kegiatan merupakan hal yang penting dilakukan agar kegiatan tersebut berlangsung secara lancar. Hal ini sangat diperlukan demi terlaksananya proses belajar mengajar yang efektif bagi sebuah jurusan di universitas tersebut (Puspaningrum dkk., 2013). Dalam penyusunan jadwal kuliah ini tentu diperlukan komponen-komponen yang mendukung agar tercipta suatu penjadwalan yang baik. Komponen-komponen tersebut bisa juga dikatakan sebagai sumber daya yang harus dipenuhi, antara lain waktu, tempat, orang, dll (Suhartono, 2015). Jika komponen tersebut telah dipenuhi maka penjadwalan tersebut bisa dikatakan baik, karena penjadwalan tersebut bisa diikuti oleh seluruh pihak yang terlibat dalam kegiatan belajar mengajar, baik dosen maupun mahasiswa.

Masalah penjadwalan dalam universitas merupakan salah satu masalah kompleks yang bisa dikaitkan dengan masalah optimasi. Hal ini dikarenakan perlunya suatu perhitungan yang cukup rumit dengan data yang banyak agar didapatkan solusi yang mendekati optimum. Namun kenyataannya proses pencarian solusi pada permasalahan ini memakan waktu komputasi yang lama. Terlebih lagi harus memenuhi tiap komponen yang ada dan komponen tiap universitas hampir berbeda. Komponen tersebut harus saling mendukung satu sama lain agar tercipta suatu penjadwalan yang baik, sebagai contoh kebutuhan mahasiswa dalam perkuliahan sebisa mungkin jangan sampai terkendala hanya karena tidak bisa mengambil

matakuliah wajib dikarenakan jadwal yang bentrok dengan matakuliah lain. Selain itu dari segi dosen, jangan sampai terjadi bentrok dengan matakuliah lain yang diajarkan (Puspaningrum dkk., 2013). Sehinga dibutuhkan suatu batasan dalam permasalahan penjadwalan agar jelas dan mudah untuk menyusunnya. Batasan tersebut terdiri dari *hard constraint*, yaitu batasan yang harus dipenuhi dan *soft constraint* yaitu batasan yang tidak harus dipenuhi namun tetap dijadikan acuan dalam penyusunannya (Mawaddah, 2006). Salah satu permasalahan penjadwalan yang kompleks adalah penjadwalan praktikum, dimana dalam permasalahan ini memiliki komponen yang harus dipenuhi yaitu waktu dosen pengampu, waktu asisten, waktu mahasiswa, jumlah ruang praktikum atau laboratorium (lab), dan masih banyak lagi.

Persoalan optimasi adalah persoalan yang menuntut pencarian solusi optimum (Marwana, 2012). Ada dua metode dalam penyelesaian masalah optimasi, yaitu (1) Metode Konvensional dan (2) Metode Heuristik. Dimana salah satu algoritma dari metode heuristik yang sering digunakan dalam permasalahan optimasi adalah *Particle Swarm Optimization* (PSO). Algoritma PSO berfokus pada penyelesaian masalah optimasi dalam pencarian ruang untuk mendapatkan solusi. Algoritma PSO dapat menyelesaikan masalah penjadwalan praktikum dengan meminimalkan kesenjangan waktu dan memaksimalkan pemanfaatan sumber daya dan *constraint* dalam penggunaan ruangan yang efektif (Mansur dkk., 2014). Pada penelitian sebelumnya yang berjudul *Particle Swarm Optimization* Untuk Sistem Informasi Penjadwalan *Resource* Di Perguran Tinggi oleh Mansur dkk., menyatakan bahwa analisa data *resource* dengan memperhatikan *constraint* mampu menghasilkan solusi yang optimal dalam penggunaan ruangan. Selain itu penelitian lain berjudul Optimasi Penjadwalan Matakuliah Di Jurusan Teknik Informatika PENS Dengan Menggunakan Algoritma *Particle Swarm Optimization* (PSO) oleh Dian Ariani dkk. berpendapat bahwa Algoritma PSO dapat digunakan untuk mengoptimasi masalah penjadwalan matakuliah di Jurusan Teknik Informatika PENS. Lalu penelitian dengan judul Penyelesaian Penjadwalan Kuliah Menggunakan Algoritma *Particle Swarm Optimization* dan *Hybrid Dimension Association Rule* oleh Ling Ria Sukmana Putri dkk., menyatakan bahwa Algoritma PSO dapat diimplementasikan pada permasalahan ini namun masih belum dapat mengatasi pelanggaran *soft constraint* hingga 100%, pernyataan lain bahwa hasil terbaik dalam penggunaan PSO ini dipengaruhi oleh parameter kognitif dan sosial (C_1 dan C_2), dimana nilai C_1 harus lebih besar dari nilai C_2.

Berdasarkan penjelasan sebelumnya di atas, pada penelitian kali ini akan dibahas mengenai penjadwalan praktikum matakuliah Kecerdasan Buatan dengan menggunakan *Modified Real Code Particle Swarm Optimization* pada Fakultas Ilmu Komputer Universitas Brawijaya. Permasalahan yang akan diselesaikan adalah pengelolaan komponen jadwal yang terdiri dari data dosen, data asisten, data mahasiswa, data ruangan, dan waktu praktikum. Lalu akan diselesaikan juga permasalahan dalam pembuatan jadwal praktikum untuk matakuliah Kecerdasan Buatan yang ditujukan bagi mahasiswa. Penjadawalan ini tidak boleh bentrok dengan jadwal asisten dan juga mahasiswa, dimana ruangan dan dosen juga dijadikan panutan dalam penyusunannya. Sehingga diharapkan dengan adanya Optimasi Penjadwalan Praktikum Kecerdasan Buatan dengan M-RCPSO dapat mengatasi permasalahan kompleks dan menghasilkan solusi yang mendekati optimum.

1.1 Batasan Masalah

Batasan masalah pada penelitian ini adalah:

1. Optimasi hanya dilakukan untuk penjadwalan praktikum pada matakuliah Kecerdasan Buatan.
2. Metode yang digunakan untuk proses optimasi penjadwalan praktikum adalah *Modified Real Code Particle Swarm Optimization*.
3. Bagian yang di-*Modified* adalah faktor konvergensi (λ).
4. Dataset yang digunakan terdiri dari data dosen, data asisten, data ruang/kelas, dan data waktu praktikum Kecerdasan Buatan di Fakultas Ilmu Komputer Universitas Brawijaya.

2. DASAR TEORI

2.1 Penjelasan Dataset

Data yang digunakan dalam penelitian adalah data asisten praktikum dari Laboratorium Komputasi Cerdas (KC) Fakultas Ilmu Komputer Universitas Brawijaya. Dataset yang didapat adalah 13 asisten seperti pada Tabel 1 dimana nantinya masing – masing kelas akan diampu oleh 2 asisten, 7 dosen seperti pada Tabel 2 dimana dosen pertama mengampu 3 kelas, dosen kedua mengampu 2 kelas dan lainnya masing-masing mengampu 1 kelas, 2 ruangan dan 10 kelas yang harus di ajar dimana setiap ruangan hanya dapat digunakan oleh 5 kelas seperti pada Tabel 3. Selain itu tiap ruangan hanya bisa digunakan dalam 5 sesi seperti pada Tabel 4.

Tabel 1. Data Asisten Praktikum

1	M. Syafiq
2	M. Kadafi
3	Sabrina Nurfadilla
4	Diva Kurnianingtyas
5	Nanda Putri
6	Radita Noer Pratiwi
7	Daneswara
8	Andriansyah Yusuf R.
9	Karina
10	Anandhi Tristiaratri
11	Agung Nurjaya Megantara

12	Ardiansyah S.
13	Fathor Rosi

Tabel 2. Data Dosen Pengampu

Dosen Pengampu	Kelas
Satrio Hadi Wijoyo	C
	I
Lailil Muflikhah	D
	E
	F
Ika Kusumaning Putri	J
Mochammad Hannats Hanafi l.	G
M. Ali Fauzi	H
Imam Cholissodin	B
Nurizal Dwi Priandani	A

Tabel 3. Data Kelas Praktikum

	Kelas				
R1	A	B	C	D	E
R2	F	G	H	I	J

Tabel 4. Data Sesi Praktikum

Sesi	Jam
1	07.00 – 08.40
2	08.40 – 10.20
3	10.20 – 12.00
4	12.50 – 14.30
5	14.30 – 16.10

2.2 Penjadwalan Praktikum Matakuliah

Penjadwalan dapat didefinisikan sebagai suatu pekerjaan untuk mendistribusikan sejumlah aktivitas yang melibatkan dosen, matakuliah, dan seterusnya, ke dalam satuan waktu (bisa termasuk ruangan) yang terbatas dengan batasan-batasan tertentu. Salah satu jenis penjadwalan adalah penjadwalan praktikum.

Praktikum berasal dari kata practicus (Latin) yang secara harfiah memiliki arti "Aktif" atau prattein (Yunani) yang memiliki arti "mengerjakan". Praktikum merupakan salah satu subsystem dari perkuliahan. Kegiatan praktikum memiliki struktur dan jadwal yang terencana. Tujuan praktikum yaitu untuk memberikan kesempatan kepada mahasiswa dalam memperoleh pengalaman nyata dan membuat mahasiswa menguasai suatu keterampilan. beberapa matakuliah tidak hanya dapat dipahami namun dapat lebih mudah diimplementasikan apabila disertai praktikum.

Pada perguruan tinggi terdapat beberapa matakuliah yang memiliki jadwal untuk kegiatan praktikum. Pada kegiatan praktikum beberapa dosen pengampu cenderung dibantu oleh asisten praktikum. Asisten praktikum merupakan mahasiswa yang tugasnya membimbing atau mendampingi praktikan dalam melakukan kegiatan praktikum. Asisten praktikum diberi tugas oleh dosen pengampu untuk memberikan materi praktik atau implementasi langsung mengenai teori yang telah dijelaskan sebelumnya oleh dosen pengampu.

Fakultas Ilmu Komputer di Universitas Brawijaya merupakan salah satu fakultas yang beberapa matakuliahnya memberikan kegiatan praktikum kepada mahasiswa. Salah satu matakuliah yang memiliki jadwal kegiatan praktikum yaitu Kecerdasan Buatan. Kecerdasan Buatan merupakan salah satu matakuliah yang membutuhkan lebih dari hanya sekedar penyampaian materi dari seorang dosen. Matakuliah tersebut membutuhkan jadwal kegiatan praktikum untuk membuat mahasiswa lebih paham mengenai teori yang disampaikan.

2.3 Algoritma *Modified Real Code Particle Swarm Optimization*

PSO adalah algoritma yang terinspirasi dari perilaku social alam, pergerakan dinamis, dan komunikasi pada serangga, burung, dan ikan. Pada penelitian yang dilakukan oleh Omar S. Soliman dkk. (2014) tentang klasifikasi Diabetes Mellitus menggunakan Modified RCPSO dan LS-SVM. Penerapan M-RCPSO pada penelitian ini digunakan untuk mencari nilai parameter optimal untuk digunakan pada algoritma LS-SVM yaitu nilai γ dan σ. Perbedaan algoritma M-RCPSO dengan PSO pada umumnya terletak pada adanya penambahan rumus untuk menghitung faktor konvergensi. Penelitian tersebut menjelaskan bahwa hasil akurasi yang didapatkan sangat tinggi jika dibandingkan dengan penelitian lain yang mengangkat permasalahan yang sama namun berbeda metode yaitu 97,833% dengan menggunakan 768 data pasien. Pada penelitian ini *Modified* RCPSO digunakan untuk mencari solusi optimal untuk penjadwalan ruang praktikum Kecerdasan Buatan yang harus memenuhi batasan – batasan yang telah ditentukan.

2.3.1 Inisialisasi

Pada proses inisialisasi ini dilakukan inisialisasi terhadap jumlah populasi, posisi awal partikel (X(t)), dan kecepatan awal partikel (V(t)). Pada penelitian ini, digunakan 5 data jadwal praktikum Kecerdasan Buatan dengan inisialisasi nilai random. Dimana representasi partikelnya diperlukan 2 asisten dalam 1 kelas, yang terdiri dari 10 kelas. Ukuran partikel adalah 10 kelas * 2 asisten. Setelah itu hasil inisialisasi posisi awal partikel akan dilakukan pembulatan dari bilangan *real* menjadi integer.

2.3.2 Proses Repair

Pada penelitian ini proses repair dilakukan dengan cara mengecek asprak yang belum dapat jadwal/kelas (*repair* dilakukan setiap inisialisasi awal dan *update* posisi) dan akan menggantikan asprak

yang mengajar lebih dari 2 kelas. Hal ini dilakukan agar representasi posisi partikel dapat menghasilkan solusi yang mendekati optimum. Proses repair sebenarnya adalah proses tambahan pada M-RCPSO sehingga prosesnya bisa dilakukan ataupun tidak. Namun untuk kasus penjadwalan praktikum ini proses repair sangat diperlukan, hal ini karena jika tidak dilakukan repair maka susah untuk mendapatkan *fitness* yang bernilai 1 sebab representasi partikelnya akan menjadi terlalu liar dalam arti susah untuk mencapai solusi yang optimum. Proses *repair* ini juga mencegah tidak terkontrolnya proses random partikel dalam representasi partikel tersebut. Hal ini akan menyulitkan proses pengujian, karena nilai fitness-nya masih ada kemungkinan lebih tinggi dari sebelumnya. Oleh karena itu, dengan adanya proses *repair* ini representasi partikel yang berupa bilangan real kemudian dibulatkan menjadi integer dan merepresentasikan *id* dari asisten akan menjadi tidak terlalu liar, dalam arti mudah untuk memperoleh nilai *fitness* 1 atau solusi yang optimum. Sehingga dalam proses pengujian akan terlihat parameter-parameter terbaiknya dalam mencapai solusi optimum (konvergen).

2.3.3 Evaluasi Nilai *Fitness* Tiap Partikel ($f(X_i)$)

Seperti pada algoritma evolusi, fungsi objektif mengukur seberapa dekat solusi dengan optimum, contohnya fungsi objektif mengukur performansi atau kualitas partikel. Pada penelitian ini nilai *fitness* tiap partikel dihitung melalui pengecekan *constraint*. Pencarian jadwal yang optimal juga dilakukan dengan memperhatikan batasan – batasan atau *constraints* (*hard & soft*) yang telah ditentukan. *Hard Constraint* merupakan batasan yang harus dipenuhi sedangan *Soft Constraint* merupakan batasan yang tidak harus dipenuhi namun tetap dijadikan acuan dalam penyusunannya. Berikut dijelaskan secara detail batasan tersebut.

- *Hard Constraint*
  a. Asisten hanya boleh mengajar maksimal 2 kelas (Berlebih).
  b. Asisten tidak boleh mengampu kelas pada dosen yang sama (Bentrok Dosen).
  c. Dalam 1 kelas tidak boleh diajar oleh 1 asisten (Berbeda).
  d. Semua asisten harus mendapat jadwal praktikum (AssKosong).
- *Soft Constraint*
  a. Pasangan asisten harus berbeda tiap kelasnya (Bentrok).

Berdasarkan penjelasan *constraint* sebelumnya pada kasus ini nilai fitness diambil dari banyaknya batasan yang dilanggar sehingga semakin kecil jumlah pelanggaran yang dihasilkan maka solusi yang dihasilkan akan semakin baik. Untuk tiap pelanggaran yang terjadi akan diberikan nilai 1. Agar tidak terjadi nilai fitness yang tak terhingga maka jumlah total semua pelanggaran akan ditambahkan 1 yang ditunjukkan pada Persamaan 1.

$$f = \frac{1}{1+(\sum Berlebih+\sum BentrokDosen+\sum Berbeda+\sum AssKosong)} \quad (1)$$

2.3.4 Update Pbest

Update *Pbest* dengan mencari nilai *fitness* maksimum tiap partikel dan bandingkan dengan nilai *fitness* maksimum sebelumnya. Jika $f(X_i) > f(Pbest_k)$, maka $f(Pbest_k) = X_i$. Lalu set nilai P_i sama dengan dengan lokasi dari nilai fitness maksimum X_i. Dimana P_i adalah hasil *update* dari tiap *Pbest* (Partikel *Best*).

2.3.5 Update Gbest

Update Gbest (*Global Best*) dengan mengevaluasi keseluruhan fitness Pbest dan mengambil yang paling maksimum.

2.3.6 *Update* Kecepatan Partikel

Update kecepatan partikel dengan rumus yang ditunjukkan pada Persamaan 2 dan iterasi, $t = t + 1$.

$$V_{id} = \lambda[\omega_{id}V_{id} + C_1r_1(p_{id} - X_{id}) + C_2r_2(p_{gd} - X_{id})] \quad (2)$$

Dimana,
i = indek partikel ke-i
d = indek dimensi ke-d
λ = faktor konvergensi
ω_{id} = bobot inersia partikel ke-i dimensi ke-d
C_1 = parameter kognitif
C_2 = parameter sosial
r_1 dan r_2 = angka random dengan range [0,1]
P_{id} = *Pbest* partikel ke-i dimensi ke-d
P_{gd} = *Gbest* partikel ke-g dimensi ke-d
X_{id} = Posisi partikel ke-i dimensi ke-d

Nilai faktor konvergensi (λ) dihitung dengan menggunakan Persamaan 3.

$$\lambda = \frac{2}{\left|2-C-\sqrt{C^2-4C}\right|} \quad (3)$$

Dimana, nilai C adalah hasil penjumlahan dari C_1 dan C_2. Selanjutnya nilai bobot inersia (ω) dapat dihitung dengan rumus yang ditunjukkan pada Persamaan 4.

$$\omega_{id} = 0.9 - \frac{t}{T_{max}} * 0.5 \quad (4)$$

Dimana,

Setelah itu lakukan perbaikan terhadap hasil update kecepatan partikel yang baru sesuai dengan kondisi berikut.

$$v_i = \begin{cases} V_{max} & if\ v_i > V_{max} \\ -V_{max} & if\ v_i > -V_{max} \end{cases}$$

2.3.7 Update Posisi Partikel

Melakukan update posisi partikel baru untuk digunakan pada iterasi selanjutnya hingga memenuhi kondisi berhenti yaitu hingga iterasi maksimum terpenuhi. Rumus update posisi ditunjukkan pada Persamaan 5.

$$X_{id} = X_{id} + (\omega V_{id}) \quad (5)$$

3. PERANCANGAN DAN IMPLEMENTASI

Pada tahap ini akan dibahas mengenai langkah-langkah dan rancangan yang digunakan dalam pembuatan sistem "Optimasi Penjadwalan Praktikum Kecerdasan Buatan Menggunakan *Modified Real Code Particle Swarm Optimization* (Studi Kasus Fakultas Ilmu Komputer Universitas Brawijaya)", rancangan implementasi *pseudocode* M-RCPSO sebagai berikut:

- Melakukan studi literatur mengenai algoritma *Modified Real Code Particle Swarm Optimization*.
- Menyusun solusi permasalahan dengan algoritma M-RCPSO.
- Menganalisis dan merancang sistem dengan menggunakan hasil pembelajaran pada tahap sebelumnya.
- Implementasi sistem berdasarkan analisis dan perancangan yang dilakukan.
- Melakukan uji coba dan evaluasi.

3.1 Implementasi *Pseudocode* M-RCPSO

Berdasarkan perancangan perangkat lunak yang telah dirancang sebelumnya, maka akan diuraikan mengenai implementasi *pseudocode* algoritma M-RCPSO sesuai dengan perancangan yang telah dibuat yang ditunjukkan pada Gambar 1. Sistem diimplementasikan menggunakan bahasa pemrograman PHP.

```
1   begin
2       t = 0
3       inisialisasi posisi partikel(X^t_{i,j}),
4   kecepatan(V^t_{i,j}), Pbest^t_{i,j}=X^t_{i,j}, hitung
5   fitness tiap partikel, dan Gbest^t_{g,j}
6   do
7           t = t + 1
8           hitung faktor konvergensi (λ)
9           update kecepatan v_{i,j}(t)
10          update posisi x_{i,j}(t)
11          hitung fitness tiap partikel
12          update Pbest_{i,j}(t) dan Gbest_{g,j}(t)
13  while (bukan kondisi berhenti)
14  end
15
```

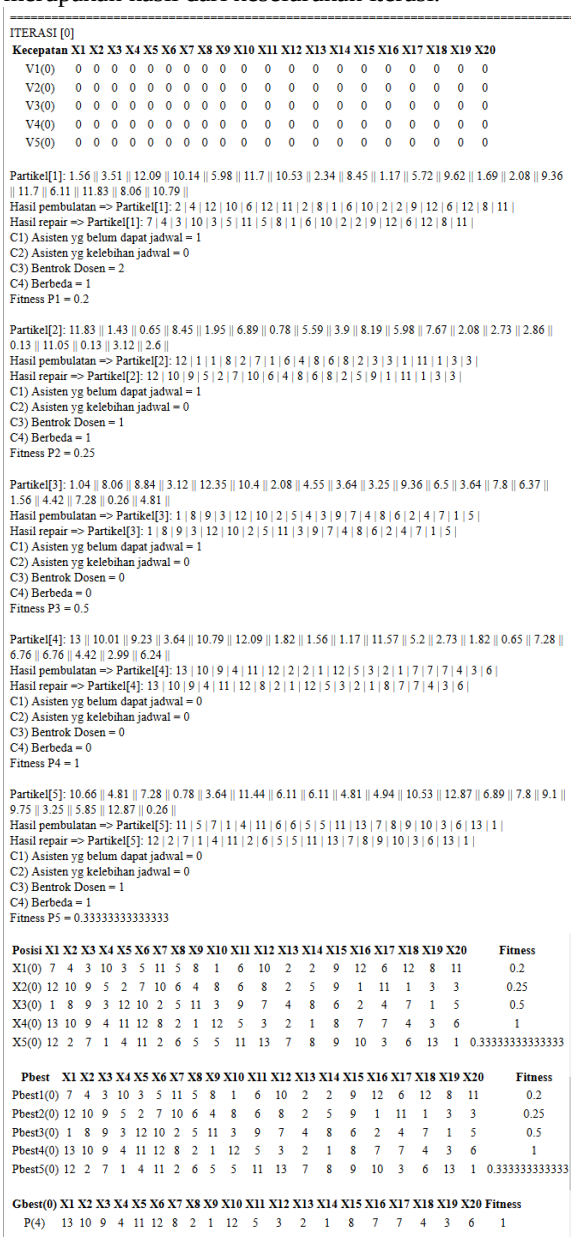
Gambar 1. *Pseudocode* Algoritma M-RCPSO

Penjelasan dari Gambar 1:

1. Baris 2 merupakan inisialisasi iterasi awal
2. Baris 3-5 merupakan proses inisialisasi posisi awal partikel, kecepatan awal partikel, *Pbest* awal partikel, evaluasi *fitness* tiap partikel, dan *Gbest* awal partikel
3. Baris 6 melakukan perulangan
4. Baris 7 inisialisasi iterasi selanjutnya
5. Baris 8-12 merupakan proses perhitungan yang ada pada algoritma M-RCPSO
6. Baris 13 bernilai benar akan melanjutkan perulangan dan bernilai salah akan keluar dari perulangan

3.2 Implementasi Antarmuka

Implementasi antarmuka berupa tampilan halaman hasil proses M-RCPSO dengan jumlah populasi partikel 5, maksimal iterasi 2, dan nilai C_1 dan C_2 berturut-turut adalah 1. Berikut hasil implementasinya yang ditunjukkan pada Gambar 2 yang merupakan hasil iterasi 0, Gambar 3 yang merupakan hasil iterasi 1, dan Gambar 4 yang merupakan hasil dari keseluruhan iterasi.


ITERASI [0]

Kecepatan X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12 X13 X14 X15 X16 X17 X18 X19 X20

V1(0) 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
V2(0) 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
V3(0) 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
V4(0) 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
V5(0) 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Partikel[1]: 1.56 || 3.51 || 12.09 || 10.14 || 5.98 || 11.7 || 10.53 || 2.34 || 8.45 || 1.17 || 5.72 || 9.62 || 1.69 || 2.08 || 9.36 || 11.7 || 6.11 || 11.83 || 8.06 || 10.79 ||
Hasil pembulatan => Partikel[1]: 2 | 4 | 12 | 10 | 6 | 12 | 11 | 2 | 8 | 1 | 6 | 10 | 2 | 2 | 9 | 12 | 6 | 12 | 8 | 11 |
Hasil repair => Partikel[1]: 7 | 4 | 3 | 10 | 3 | 5 | 11 | 5 | 8 | 1 | 6 | 10 | 2 | 2 | 9 | 12 | 6 | 12 | 8 | 11 |
C1) Asisten yg belum dapat jadwal = 1
C2) Asisten yg kelebihan jadwal = 0
C3) Bentrok Dosen = 2
C4) Berbeda = 1
Fitness P1 = 0.2

Partikel[2]: 11.83 || 1.43 || 0.65 || 8.45 || 1.95 || 6.89 || 0.78 || 5.59 || 3.9 || 8.19 || 5.98 || 7.67 || 2.08 || 2.73 || 2.86 || 0.13 || 11.05 || 0.13 || 3.12 || 2.6 ||
Hasil pembulatan => Partikel[2]: 12 | 1 | 1 | 8 | 2 | 7 | 1 | 6 | 4 | 8 | 6 | 8 | 2 | 3 | 3 | 1 | 11 | 1 | 3 | 3 |
Hasil repair => Partikel[2]: 12 | 10 | 9 | 5 | 2 | 7 | 10 | 6 | 4 | 8 | 6 | 8 | 2 | 5 | 9 | 1 | 11 | 1 | 3 | 3 |
C1) Asisten yg belum dapat jadwal = 1
C2) Asisten yg kelebihan jadwal = 0
C3) Bentrok Dosen = 1
C4) Berbeda = 1
Fitness P2 = 0.25

Partikel[3]: 1.04 || 8.06 || 8.84 || 3.12 || 12.35 || 10.4 || 2.08 || 4.55 || 3.64 || 3.25 || 9.36 || 6.5 || 3.64 || 7.8 || 6.37 || 1.56 || 4.42 || 7.28 || 0.26 || 4.81 ||
Hasil pembulatan => Partikel[3]: 1 | 8 | 9 | 3 | 12 | 10 | 2 | 5 | 4 | 3 | 9 | 7 | 4 | 8 | 6 | 2 | 4 | 7 | 1 | 5 |
Hasil repair => Partikel[3]: 1 | 8 | 9 | 3 | 12 | 10 | 2 | 5 | 11 | 3 | 9 | 7 | 4 | 8 | 6 | 2 | 4 | 7 | 1 | 5 |
C1) Asisten yg belum dapat jadwal = 1
C2) Asisten yg kelebihan jadwal = 0
C3) Bentrok Dosen = 0
C4) Berbeda = 0
Fitness P3 = 0.5

Partikel[4]: 13 || 10.01 || 9.23 || 3.64 || 10.79 || 12.09 || 1.82 || 1.56 || 1.17 || 11.57 || 5.2 || 2.73 || 1.82 || 0.65 || 7.28 || 6.76 || 6.76 || 4.42 || 2.99 || 6.24 ||
Hasil pembulatan => Partikel[4]: 13 | 10 | 9 | 4 | 11 | 12 | 2 | 2 | 1 | 12 | 5 | 3 | 2 | 1 | 7 | 7 | 7 | 4 | 3 | 6 |
Hasil repair => Partikel[4]: 13 | 10 | 9 | 4 | 11 | 12 | 8 | 2 | 1 | 12 | 5 | 3 | 2 | 1 | 8 | 7 | 7 | 4 | 3 | 6 |
C1) Asisten yg belum dapat jadwal = 0
C2) Asisten yg kelebihan jadwal = 0
C3) Bentrok Dosen = 0
C4) Berbeda = 0
Fitness P4 = 1

Partikel[5]: 10.66 || 4.81 || 7.28 || 0.78 || 3.64 || 11.44 || 6.11 || 6.11 || 4.81 || 4.94 || 10.53 || 12.87 || 6.89 || 7.8 || 9.1 || 9.75 || 3.25 || 5.85 || 12.87 || 0.26 ||
Hasil pembulatan => Partikel[5]: 11 | 5 | 7 | 1 | 4 | 11 | 6 | 6 | 5 | 5 | 11 | 13 | 7 | 8 | 9 | 10 | 3 | 6 | 13 | 1 |
Hasil repair => Partikel[5]: 12 | 2 | 7 | 1 | 4 | 11 | 2 | 6 | 5 | 5 | 11 | 13 | 7 | 8 | 9 | 10 | 3 | 6 | 13 | 1 |
C1) Asisten yg belum dapat jadwal = 0
C2) Asisten yg kelebihan jadwal = 0
C3) Bentrok Dosen = 1
C4) Berbeda = 1
Fitness P5 = 0.333333333333333

Posisi X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12 X13 X14 X15 X16 X17 X18 X19 X20 Fitness

X1(0) 7 4 3 10 3 5 11 5 8 1 6 10 2 2 9 12 6 12 8 11 0.2
X2(0) 12 10 9 5 2 7 10 6 4 8 6 8 2 5 9 1 11 1 3 3 0.25
X3(0) 1 8 9 3 12 10 2 5 11 3 9 7 4 8 6 2 4 7 1 5 0.5
X4(0) 13 10 9 4 11 12 8 2 1 12 5 3 2 1 8 7 7 4 3 6 1
X5(0) 12 2 7 1 4 11 2 6 5 5 11 13 7 8 9 10 3 6 13 1 0.333333333333333

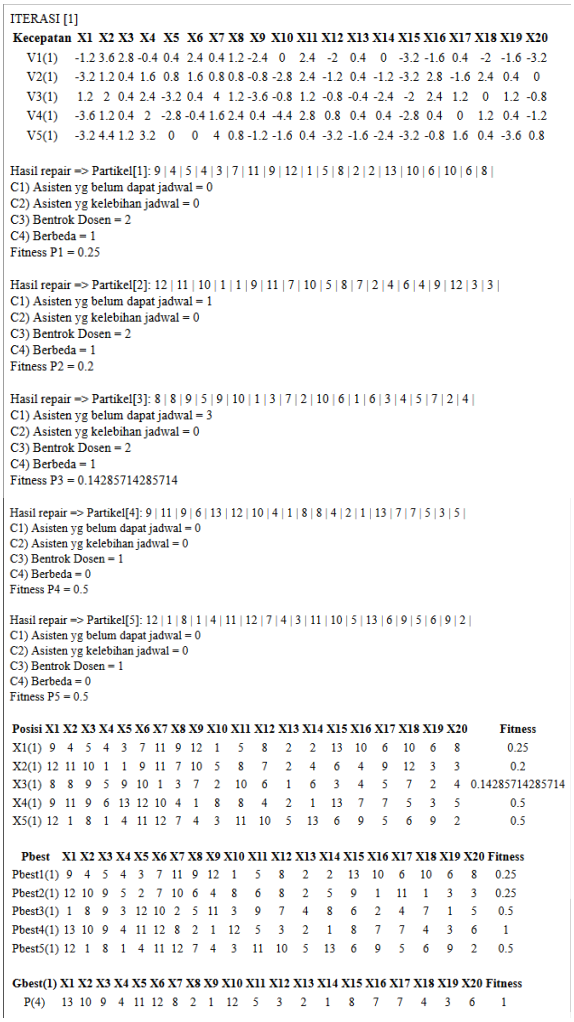
Pbest X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12 X13 X14 X15 X16 X17 X18 X19 X20 Fitness

Pbest1(0) 7 4 3 10 3 5 11 5 8 1 6 10 2 2 9 12 6 12 8 11 0.2
Pbest2(0) 12 10 9 5 2 7 10 6 4 8 6 8 2 5 9 1 11 1 3 3 0.25
Pbest3(0) 1 8 9 3 12 10 2 5 11 3 9 7 4 8 6 2 4 7 1 5 0.5
Pbest4(0) 13 10 9 4 11 12 8 2 1 12 5 3 2 1 8 7 7 4 3 6 1
Pbest5(0) 12 2 7 1 4 11 2 6 5 5 11 13 7 8 9 10 3 6 13 1 0.333333333333

Gbest(0) X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12 X13 X14 X15 X16 X17 X18 X19 X20 Fitness

P(4) 13 10 9 4 11 12 8 2 1 12 5 3 2 1 8 7 7 4 3 6 1

Gambar 2. Antarmuka Hasil Proses M-RCPSO Iterasi 0



ITERASI [1]

Kecepatan	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16	X17	X18	X19	X20
V1(1)	-1.2	3.6	2.8	-0.4	0.4	2.4	0.4	1.2	-2.4	0	2.4	-2	0.4	0	-3.2	-1.6	0.4	-2	-1.6	-3.2
V2(1)	-3.2	1.2	0.4	1.6	0.8	1.6	0.8	0.8	-0.8	-2.8	2.4	-1.2	0.4	-1.2	-3.2	2.8	-1.6	2.4	0.4	0
V3(1)	1.2	2	0.4	2.4	-3.2	0.4	4	1.2	-3.6	-0.8	1.2	-0.8	-0.4	-2.4	-2	2.4	1.2	0	1.2	-0.8
V4(1)	-3.6	1.2	0.4	2	-2.8	-0.4	1.6	2.4	0.4	-4.4	2.8	0.8	0.4	0.4	-2.8	0.4	0	1.2	0.4	-1.2
V5(1)	-3.2	4.4	1.2	3.2	0	0	4	0.8	-1.2	-1.6	0.4	-3.2	-1.6	-2.4	-3.2	-0.8	1.6	0.4	-3.6	0.8

Hasil repair => Partikel[1]: 9 | 4 | 5 | 4 | 3 | 7 | 11 | 9 | 12 | 1 | 5 | 8 | 2 | 2 | 13 | 10 | 6 | 10 | 6 | 8 |
C1) Asisten yg belum dapat jadwal = 0
C2) Asisten yg kelebihan jadwal = 0
C3) Bentrok Dosen = 2
C4) Berbeda = 1
Fitness P1 = 0.25

Hasil repair => Partikel[2]: 12 | 11 | 10 | 1 | 1 | 9 | 11 | 7 | 10 | 5 | 8 | 7 | 2 | 4 | 6 | 4 | 9 | 12 | 3 | 3 |
C1) Asisten yg belum dapat jadwal = 1
C2) Asisten yg kelebihan jadwal = 0
C3) Bentrok Dosen = 2
C4) Berbeda = 1
Fitness P2 = 0.2

Hasil repair => Partikel[3]: 8 | 8 | 9 | 5 | 9 | 10 | 1 | 3 | 7 | 2 | 10 | 6 | 1 | 6 | 3 | 4 | 5 | 7 | 2 | 4 |
C1) Asisten yg belum dapat jadwal = 3
C2) Asisten yg kelebihan jadwal = 0
C3) Bentrok Dosen = 2
C4) Berbeda = 1
Fitness P3 = 0.14285714285714

Hasil repair => Partikel[4]: 9 | 11 | 9 | 6 | 13 | 12 | 10 | 4 | 1 | 8 | 8 | 4 | 2 | 1 | 13 | 7 | 7 | 5 | 3 | 5 |
C1) Asisten yg belum dapat jadwal = 0
C2) Asisten yg kelebihan jadwal = 0
C3) Bentrok Dosen = 1
C4) Berbeda = 0
Fitness P4 = 0.5

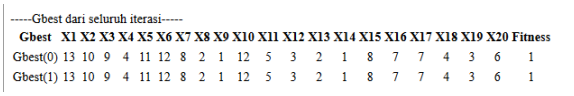
Hasil repair => Partikel[5]: 12 | 1 | 8 | 1 | 4 | 11 | 12 | 7 | 4 | 3 | 11 | 10 | 5 | 13 | 6 | 9 | 5 | 6 | 9 | 2 |
C1) Asisten yg belum dapat jadwal = 0
C2) Asisten yg kelebihan jadwal = 0
C3) Bentrok Dosen = 1
C4) Berbeda = 0
Fitness P5 = 0.5

Posisi	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16	X17	X18	X19	X20	Fitness
X1(1)	9	4	5	4	3	7	11	9	12	1	5	8	2	2	13	10	6	10	6	8	0.25
X2(1)	12	11	10	1	1	9	11	7	10	5	8	7	2	4	6	4	9	12	3	3	0.2
X3(1)	8	8	9	5	9	10	1	3	7	2	10	6	1	6	3	4	5	7	2	4	0.142857142857143
X4(1)	9	11	9	6	13	12	10	4	1	8	8	4	2	1	13	7	7	5	3	5	0.5
X5(1)	12	1	8	1	4	11	12	7	4	3	11	10	5	13	6	9	5	6	9	2	0.5

Pbest	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16	X17	X18	X19	X20	Fitness
Pbest1(1)	9	4	5	4	3	7	11	9	12	1	5	8	2	2	13	10	6	10	6	8	0.25
Pbest2(1)	12	10	9	5	2	7	10	6	4	8	6	8	2	5	9	1	11	1	3	3	0.25
Pbest3(1)	1	8	9	3	12	10	2	5	11	3	9	7	4	8	6	2	4	7	1	5	0.5
Pbest4(1)	13	10	9	4	11	12	8	2	1	12	5	3	2	1	8	7	7	4	3	6	1
Pbest5(1)	12	1	8	1	4	11	12	7	4	3	11	10	5	13	6	9	5	6	9	2	0.5

Gbest(1)	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16	X17	X18	X19	X20	Fitness
P(4)	13	10	9	4	11	12	8	2	1	12	5	3	2	1	8	7	7	4	3	6	1

Gambar 3. Antarmuka Hasil Proses M-RCPSO Iterasi 1



-----Gbest dari seluruh iterasi-----

Gbest	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16	X17	X18	X19	X20	Fitness
Gbest(0)	13	10	9	4	11	12	8	2	1	12	5	3	2	1	8	7	7	4	3	6	1
Gbest(1)	13	10	9	4	11	12	8	2	1	12	5	3	2	1	8	7	7	4	3	6	1

Gambar 4. Hasil Gbest Keseluruhan Iterasi

4. PENGUJIAN DAN ANALISIS

4.1 Pengujian Ukuran Populasi (*PopSize*)

Pada pengujian ukuran populasi (*PopSize*) dilakukan untuk mengetahui ukuran populasi yang dapat menghasilkan solusi paling optimal pada kasus penjadwalan praktikum matakuliah Kecerdasan Buatan Fakultas Ilmu Komputer. Pengujian populasi ini akan dilakukan sebanyak 5 kali dengan jumlah iterasi sebanyak 5. Ukuran populasi yang akan diuji merupakan nilai kelipatan 5 sampai dengan 50 dan kombinasi C_1 dan C_2 adalah 1. Setelah diketahui nilai *fitness*nya lalu dilakukan rata-rata terhadap nilai *fitness* tersebut, dari situ dapat diketahui nilai rata-rata *fitness* terbaik berdasarkan pengujian populasi ini adalah 1 dan didapatkan mulai dari pengujian dengan banyak populasi 10 sampai dengan 50, dimana semakin besar nilai *fitness* (mendekati 1) maka semakin mendekati solusi optimum. Berikut hasil dari pengujian populasi (*PopSize*) yang ditunjukkan pada Tabel 5.

Tabel 5. Hasil Pengujian Ukuran Populasi (*PopSize*)

Banyak *Popsize*	Nilai *Fitness* Percobaan ke-					Rata-rata nilai *Fitness*
	1	2	3	4	5	
5	0,5	1	1	1	1	0,9
10	1	1	1	1	1	1
15	1	1	1	1	1	1
20	1	1	1	1	1	1
25	1	1	1	1	1	1
30	1	1	1	1	1	1
35	1	1	1	1	1	1
40	1	1	1	1	1	1
45	1	1	1	1	1	1
50	1	1	1	1	1	1

4.2 Pengujian Banyaknya Iterasi

Uji coba banyak iterasi pada Tabel 6 akan dilakukan 5 kali dengan menggunakan kombinasi parameter kognitif dan sosial C_1 dan C_2 dan banyaknya *popsize* yang digunakan adalah *popsize* terbaik dari pengujian ukuran populasi yang sebelumnya telah dilakukan yaitu 10. Banyaknya iterasi yang akan diuji merupakan nilai kelipatan 10 sampai dengan 100. Setelah diketahui nilai *fitness*nya lalu dilakukan rata-rata terhadap nilai *fitness* tersebut, dari hal itu dapat diketahui nilai rata-rata *fitness* terbaik berdasarkan pengujian iterasi ini adalah 1 dan didapatkan mulai dari pengujian dengan banyak iterasi 10.

Tabel 6. Hasil Pengujian Banyaknya Iterasi

Banyak Iterasi	Nilai *Fitness* Percobaan ke-					Rata-rata nilai *Fitness*
	1	2	3	4	5	
10	1	1	1	1	1	1
20	1	1	1	1	1	1
30	1	1	1	1	1	1
40	1	1	1	1	1	1
50	1	1	1	1	1	1
60	1	1	1	1	1	1
70	1	1	1	1	1	1
80	1	1	1	1	1	1
90	1	1	1	1	1	1
100	1	1	1	1	1	1

4.3 Pengujian Kombinasi Parameter Kognitif dan Sosial (C_1 dan C_2)

Pengujian kombinasi C_1 dan C_2 yang dilakukan bertujuan untuk mendapatkan kombinasi C_1 dan C_2 yang dapat menghasilkan nilai *fitness* paling besar. Uji coba dilakukan sebanyak 5 kali dengan kenaikan kombinasi probabilitasnya sebanyak 0.2 dalam rentang 2.5-0,5 untuk C_1 dan rentang 0,5-2,5 untuk C_2. Dimana C_1 dan C_2 dicari dengan menggunakan rumus *time variant* yang ditunjukkan pada Persamaan 6 dan Persamaan 7. .Pengujian ini dilakukan menggunakan jumlah *popsize* terbaik hasil pengujian *popsize* yaitu 10 dan jumlah iterasi hasil pengujian iterasi sebelumnya yaitu 10. Hasil pengujian kombinasi C_1 dan C_2 terbaik terdapat pada kombinasi C_1 mulai dari 2.3 sampai 0.5 dan C_2 mulai dari 0.7 sampai 2.5 pada permasalahan penjadwalan asisten praktikum matakuliah Kecerdasan Buatan. Berikut hasil dari pengujian kombinasi C_1 dan C_2 yang ditunjukkan pada Tabel 7.

$$C_1 = \left(C_{1f} - C_{1i}\right)\frac{t}{t_{max}} + C_{1i} \quad (6)$$

$$C_2 = \left(C_{2f} - C_{2i}\right)\frac{t}{t_{max}} + C_{2i} \quad (7)$$

Tabel 7. Hasil Pengujian Parameter Kognitif dan Sosial (C_1 dan C_2)

Kombinasi		Nilai *fitness* percobaan generasi ke-					Rata-rata *fitness*
C_1	C_2	1	2	3	4	5	
2.3	0.7	1	1	1	1	1	1
2.1	0.9	1	1	1	1	1	1
1.9	1.1	1	1	1	1	1	1
1.7	1.3	1	1	1	1	1	1
1.5	1.5	1	1	1	1	1	1
1.3	1.7	1	1	1	1	1	1
1.1	1.9	1	1	1	1	1	1
0.9	2.1	1	1	1	1	1	1
0.7	2.3	1	1	1	1	1	1
0.5	2.5	1	1	1	1	1	1

4.4 Pengujian Parameter Terbaik

Uji coba ini menggunakan parameter terbaik yang telah diperoleh dari pengujian sebelumnya yaitu jumlah popsize sebanyak 10, jumlah iterasi sebanyak 10 hal ini dikarenakan pada iterasi pertama fitness yang didapatkan telah optimum hingga iterasi maksimum tercapai, nilai C_1 = 2.3 dan C_2 = 0.7 untuk mendapatkan hasil penjadwalan praktikum matakuliah Kecerdasan Buatan seperti pada Tabel 8.

Tabel 8. Hasil Pengujian Parameter Terbaik

R1	Lailil		Lailil		Lailil		Satrio		Satrio	
	Ardi	Agung	Anan	Fathor	Syafiq	Sabrina	Kadafi	Danes	Nanda	Syafiq
R2	Ika		Hannats		Ali		Imam		Nurizal	
	Radita	Karina	Sabrina	Danes	Radita	Andri	Diva	Kadafi	Andri	Anan

4.5 Analisis Hasil Pengujian

Berdasarkan pengujian yang telah dilakukan, dapat disimpulkan bahwa sistem telah memberikan solusi yang optimal dengan besar *fitness* 1 dimana *fitness* 1 menandakan bahwa solusi yang dihasilkan telah memenuhi syarat yang telah ditentukan, namun dalam hal ini sistem mencapai global optimal dimana solusi optimal didapatkan secara cepat pada iterasi – iterasi awal. Hal ini disebabkan oleh masih kecilnya ruang lingkup pada permasalahan penjadwalan ini dan juga karena adanya proses repair yang diterapkan pada sistem.

5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil uji coba parameter algoritma Optimasi Penjadwalan Praktikum Kecerdasan Buatan Menggunakan *Modified Real Code Particle Swarm Optimization* dengan Studi Kasus Fakultas Ilmu Komputer Universitas Brawijaya, terdapat beberapa kesimpulan yaitu:

1. Untuk mengukur solusi dari permasalahan optimasi M-RCPSO ini dilakukan proses *repair*, dimana cek asisten yang jadwalnya > 2 dan di-*repair* dengan asisten yang belum mendapat jadwal.
2. Dari hasil pengujian ukuran populasi sebanyak 10 dengan nilai rata-rata *fitness* tertingginya yaitu 1, banyaknya iterasi yaitu 10 dengan nilai rata-rata *fitness* tertingginya 1 , kombinasi C_1 dan C_2 sebesar 2.3 sampai 0.5 dan C_2 mulai dari 0.7 sampai 2.5 dengan nilai rata-rata *fitness* tertingginya 1 dan *fitness* terbaik hal ini menunjukkan bahwa solusi yang didapatkan sudah mendekati optimum.

5.2 Saran

Berdasarkan beberapa kesimpulan yang telah didapatkan, sistem optimasi penjadwalan praktikum Kecerdasan Buatan dengan M-RCPSO memiliki beberapa kekurangan sehingga dibu-tuhkan beberapa saran untuk pengembangan sistem ke penelitian selanjutnya antara lain:

1. Ruang lingkup dalam optimasi penjadwalan ini masih pada tingkat jurusan, sehingga dapat diperbesar sampai ketingkat universitas.
2. Proses pengujian diharapkan dapat menguji ke semua aspek variabel yang ada dalam algoritma sehingga dapat mengetahui tingkat fitness secara akurat dan lebih relevan.
3. Optimasi penjadwalan ini bisa lebih berkembang dengan menambahkan lebih banyak constraint.

4. Perlu dilakukan pencegahan konvergensi dini dengan menggunakan algoritma lain seperti *random injection*.

6. DAFTAR PUSTAKA

ARIANI, DIAN, dkk., 2011. Optimasi Penjadwalan Mata Kuliah Di Jurusan Teknik Informatika PENS Dengan Menggunakan Algoritma Particle Swarm Optimization (PSO). pp. 1-11.

MANSUR, dkk., 2014. Particle Swarm Optimization Untuk Sistem Informasi Penjadwalan Resource Di Perguruan Tinggi. Jurnal Sistem Informasi Bisnis, pp. 11-19.

MAWADDAH, NIA KURNIA. 2006. Optimasi Penjadwalan Ujian Menggunakan Algoritma Genetika. Vol 2, No 2, Hal 1-8. Universitas Brawijaya Malang.

PUSPANINGRUM, WIGA AYU, dkk. 2013. Penjadwalan Mata Kuliah Menggunakan Algoritma Genetika di Jurusan Sistem Informasi ITS. Jurnal Teknik Pomits. Vol 2, No 1, Hal 127-131. Institut Teknologi Sepuluh November.

PUTRI, LING RIA SUKMANA, dkk., 2013. Penyelesaian Penjadwalan Kuliah Menggunakan Algoritma Particle Swarm Optimization and Hybrid Dimension Association Rule.

SOLIMAN, OMAR S., dkk., 2014. Classification of Diabetes Mellitus using Modified Particle Swarm Optimization and Least Squares Support Vector Machine. International Journal of Computer and Technology (IJCTT), vol. 8, no. 1, Februari 2014, pp. 38-44.

SUHARTONO, ENTOT. 2015. Optimasi Penjadwalan Mata Kuliah dengan Algoritma Genetika (Studi Kasus di AMIK JTC Semarang). No 2, September 2015, Hal 132-146. AMIK JTC Semarang.

Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) p-ISSN: 2355-7699
Vol. 3, No. 4, Desember 2016, hlm. 273-278 e-ISSN: 2528-6579

IMPLEMENTASI METODE NAÏVE BAYES CLASSIFIER UNTUK SELEKSI ASISTEN PRAKTIKUM PADA SIMULASI HADOOP MULTINODE CLUSTER

Maryamah[1], Moh. Fadel Asikin[2], Daisy Kurniawaty[3], Selly Kurnia Sari[4], Imam Cholissodin[5]

[1,2,3,4,5]Fakultas Ilmu Komputer Universitas Brawijaya
Email: [1]maryamahfaisol02@gmail.com, [2]fadelasikin@gmail.com, [3]daisyniaa@gmail.com, [4]sellykurniasr@gmail.com, [5]imamcs@ub.ac.id

(Naskah masuk: 1 Oktober 2016, diterima untuk diterbitkan: 26 Desember 2016)

Abstrak

Pemilihan asisten pada praktikum dialami oleh berbagai universitas di Indonesia salah satunya di Fakultas Ilmu Komputer (FILKOM) Universitas Brawijaya (UB). Dalam pemilihan asisten praktikum ada beberapa proses yang harus dilalui. Proses pemilihan yang ada di FILKOM masih dalam bentuk manual. Adapun proses yang dijalani dalam pemilihan asisten praktikum diantaranya adalah tes administrasi, tes *live coding*, dan tes mengajar. Dalam penentuan penerimaan asisten praktikum tersebut berdasarkan hasil tes yang telah dilakukan. Kendala yang dihadapi adalah kemiripan hasil tes pada ketentuan tertentu yang menyebabkan kerancuan proses pemilihan asisten praktikum. Dari permasalahan tersebut penulis melihat suatu peluang untuk membuat sistem yang mampu menyeleksi dengan cara melakukan hasil klasifikasi tertinggi terhadap hasil tes yang dilakukan oleh calon asisten praktikum. Penelitian ini menggunakan *Hadoop* dengan menerapkan metode *Naïve Bayes* yang diangap mampu menghasilkan klasifikasi yang akurat, sehingga dapat mempermudah seorang dosen dalam memilih asisten praktikum dengan kualitas yang baik.

Kata kunci: *big data, klasifikasi, naïve bayes, hadoop*

Abstract

Selection of the lab assistants experienced by various universities in Indonesia one of them in the Faculty of Computer Science (FILKOM) University of Brawijaya (UB). In the selection of lab assistant, there is some process to be followed. The election process is in FILKOM still in manual form. The process according to which the election is a test lab assistant include administration, test live coding, and test of teaching. In determining the lab assistant acceptance is based on the tests that have been carried out. Obstacles encountered is the similarity test results on the specific provisions that caused confusion electoral process lab assistant. Of these issues, the authors saw an opportunity to create a system that is able to select a way to the highest classification results against the results of tests conducted by the lab assistant candidates. This study uses Hadoop with Naïve Bayes applying methods as may be capable of producing an accurate classification, so as to facilitate a lecturer in choosing a lab assistant with good quality.

Keywords: *big data, classification, naïve bayes, hadoop*

1. PENDAHULUAN

Perkembangan teknologi saat ini membawa kita pada perubahan hampir dari semua aspek di kehidupan. Dalam menentukan sesuatu, manusia pasti dihadapkan banyak pilihan. Dengan adanya pilihan yang benar, akan sangat mempengaruhi kehidupan manusia itu sendiri. Seiring berkembangnya teknologi, perlu dikembangkan dan juga ditingkatkan suatu sumber daya manusia agar semakin berkualitas. Permasalahan seleksi menentukan asisten praktikum banyak dialami di berbagai universitas di Indonesia, khususnya di Fakultas Ilmu Komputer (Filkom) Universitas Brawijaya.

Proses seleksi pada asisten praktikum, di Filkom masih menggunakan cara yang manual. Proses pemilihan asisten praktikum dilakukan dengan menjalani beberapa tes yang sudah ditentukan, yaitu melalui tiga tahap tes diantaranya tes administrasi, tes *live coding*, dan tes mengajar. Untuk penentuannya dilihat dari hasil tes yang sudah dilakukan sebelumnya. Kemiripan hasil nilai tes pada ketentuan tertentu dapat mengakibatkan kerancuan pada proses pemilihan asisten praktikum, serta dapat berpengaruh pada kualitasnya. Dan sangat diharapkan agar pemilihan asisten praktikum terpilih secara tepat. Untuk menghindari kerancuan hasil, sangat dibutuhkannya suatu sistem yang mampu membantu dalam seleksi pemilihan asisten praktikum baru dengan hasil seleksi yang tepat.

Pada penelitian oleh (Kalamsyah, 2014) yaitu tentang aplikasi untuk pemilihan asisten praktikum dan juga lab yang berbasis web menggunakan metode pengerjaan SDLC *waterfall* pada tahap implementasi. Orang yang terlibat dalam aplikasi tersebut adalah seorang laboran sebagai admin yang

mempunyai akses penuh dalam mengelola infomasi seleksi, melakukan *input* nilai, dan melihat hasil perangkingan. Untuk perhitungan seleksi menggunakan metode *Analytic Hierarchi Process*. Hasil dari penelitian tersebut, laboran dapat mengetahui nilai paling tinggi sampai nilai paling rendah dari semua hasil seleksi. Hasil seleksi di pilih dengan nilai tes asisten baru dengan nilai tertinggi.

Selanjutnya penelitian yang dilakukan oleh (Sholihah, 2013), membahas mengenai penentuan jumlah UKT di Universitas Trunojoyo Madura. Penentuan UKT tersebut berdasarkan kriteria yang sesuai. Dalam pembuatan sistem, peneliti menggunakan metode Naïve Bayes Classifier (NBC). Hasil yang didapatkan pada skenario satu yaitu 90%, dan skenario hasil dua data kategori 78%.

Kemudian penelitian Naive Bayes Classifier oleh (Kusumadewi, 2009) adalah penggunaan alat ukur antropometri sebagai variabel masukan dalam pengklasifikasian status gizi manusia. Untuk mendapatkan hasil yang optimal terdapat beberapa pedoman antropometri yaitu usia dan berat badan, tinggi dan panjang badan, serta lingkar lengan atas. Dalam penelitian ini didapatkan 5 golongan untuk status gizi, diantaranya berat kurang, berat normal, obesitas kurang, obesitas sedang, dan obesitas berat. Untuk klasifikasi status gizi metode yang digunakan adalah metode *Naive Bayes Classiffier* dengan hasil akurasi cukup tinggi yaitu 93,2%.

Berdasarkan permasalahan di atas, maka dibuat suatu sistem yang mampu menyeleksi dengan cara melihat hasil klasifikasi tertinggi terhadap hasil tes yang dilakukan oleh calon asisten praktikum. Penelitian ini menggunakan konsep *Big Data* yang seiring berjalannya waktu data yang digunakan akan semakin besar sebagai data histori yang dikumpulkan. Pada konsep tersebut menggunakan *Hadoop* yang dapat memproses data dalam ukuran besar. Metode yang digunakan adalah *Naïve Bayes* karena dianggap mampu menghasilkan klasifikasi yang akurat, sehingga dapat mempermudah seorang dosen dalam memilih asisten praktikum dengan kualitas yang baik.

2. DASAR TEORI

2.1 Big Data

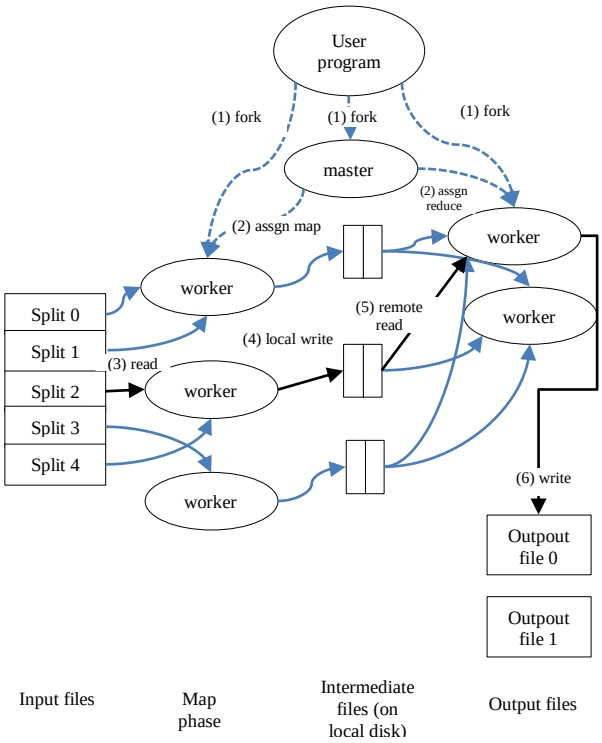
Konsep *Big Data* pada dasarnya adalah sebuah "lautan data", dengan banyaknya informasi dan sarana untuk menganalisanya (Trivu dan Ivan, 2014). *Big Data* merupakan sebuah kombinasi teknologi yang dapat mengelola data yang beragam dalam jumlah besar, dengan kecepatan dan waktu yang tepat dalam keperluan analisis, dan pada saat yang tepat untuk keperluan analisis dan reaksi. *Big Data* memiliki tiga karakteristik, yaitu *volume*, *velocity*, dan *variety*.

Big Data dideskripsikan sebagai suatu dataset dan teknik analisis di dalam suatu aplikasi yang memiliki jumlah data sangat besar dan juga sangat kompleks. *Big Data* membutuhkan suatu penyimpanan, manajemen, analisis, dan juga teknologi visualisasi (Chen, Chiang, & Storey, 2012, p 1161 dalam (Irem, 2016). Menurut McAfee & Bryjonlfsson, 2012 dalam (Irem, 2016), menerangkan bahwa untuk memanggil data yang begitu besar, dibutuhkan beberapa kategori, terdapat tiga kategori yang memenuhi, diantaranya *volume*, *velocity*, dan *variety*. Dengan masing-masing penjelasan volume merupakan jumlah pada data, velocity menerangkan tentan kecepatan pencarian informasi data, kemudian variety adalah macam-macam jenis data misalnya dalam bentuk txt, doc, dan lain sebagainya.

2.1.1 Mapreduce

Mapreduce merupakan suatu *framework* yang digunakan pada aplikasi dan program yang dikenalkan oleh google untuk menjalankan pekerjaan komputasi yang terdistribusi dan dijalankan pada *cluster*. Konsep yang digunakan dalam Mapreduce adalah fungsi Map dan Reduce untuk *functional programming* (Industri, 2013).

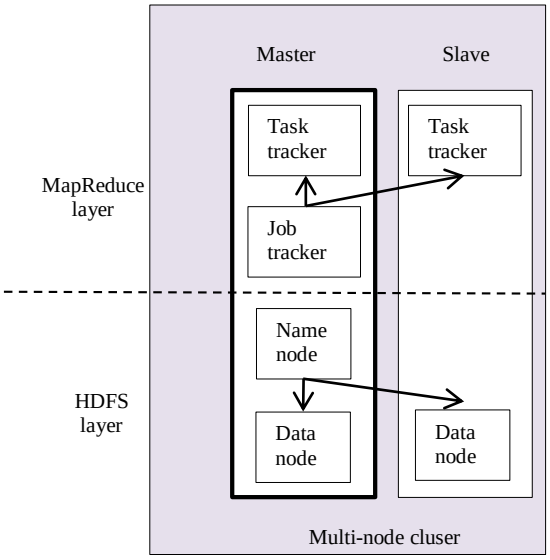
Hadoop Mapreduce termasuk bagian dari susunan kerja yang dibuat agar lebih mudah dalam melakukan suatu perhitungan. Hadoop Mapreduce ini membuat user untuk bertugas lebih ringan dan berjalan secara sejajar dalam node berbeda, menghemat waktu dalam mengeksekusi keseluruhan.



Gambar 1. Mapreduce

2.1.2 Multi Node

Dalam hadoop multi node ini dibutuhkan dua buah komputer, komputer pertama digunakan pada cluster sedangkan komputer kedua digunakan pada slave. Dua komputer ini dikonfigurasi yang merupakan dari mesin dua *single node*.



Gambar 2. Hadoop Multi Node

2.2 Klasifikasi

Klasifikasi merupakan penentuan objek ke dalam suatu kategori atau kelas. Penentuan objek menggunakan beberapa model (Han, 2006). Dalam memulai suatu klasifikasi data dengan membangun sebuah *rule* klasifikasi dengan algoritma tertentu yang digunakan pada data *training* dan data *testing.*

Pada penelitian ini untuk klasifikasi dalam perhitungannya menggunakan metode *Naïve Bayes Classifier.*

2.3 Naive Bayes Classifier

Naïve Bayes Classifier merupakan penyederhanaan dari teorema Bayes, penemu metode ini adalah seorang ilmuwan Inggris yang bernama Thomas Bayes. Algoritma dalam metode Naïve Bayes didasarkan dengan teknik klasifikasi (Kusumadewi, 2009) dapat dibuktikan bahwa saat kecepatan sangat tinggi dan bersamaan diaplikasikan dalam suatu database dengan jumlah data yang besar, naive bayes mempunyai akurasi dan juga kecepatan yang tinggi (Nugroho, 2009).

Metode *Naive Bayes*dengan prinsip teorema *Bayes* mempunyai atribut yang saling berhubungan satu sama lain. Pendekatan yang digunakan teorema bayes yaitu menghitung probabilitas sebuah kejadian pada kondisi tertentu (Lukito & Chrismanto, 2015). Dasar dari teorema *Bayes* dinyatakan dalam persamaan (Bustami, 2013).

$$P\ (H\ |X) = \frac{P(X|H).P(H)}{P(X)} \quad (1)$$

Keterangan:

X : Data kelas yang belum diketahui.

H : Hipotesis dari data *X* yaitu suatu kelas Spesifik.

P(H| X) : Probabilitas Hipotesis *H* berdasarkan kondisi *X*.

P(H) : Probabilitas Hipotesis *H*

P(X | H) : Probabilitas *X* berdasarkan kondisi *H*

P(X) : Probabilitas *X*

Pada rumus di atas dapat dijelaskan bahwa teorema naive bayes dibutuhkan sebuah petunjuk sebagai proses penentu kelas yang sesuai dengan sampel. Sehingga dibutuhkan kesesuaian terhadap teorema bayes sebagai berikut:

$$P\ (C|F1\ldots Fn) = \frac{P(C)P(F1\ldots Fn|C)}{P(F1\ldots Fn)} \quad (2)$$

Keterangan:

C : Sebagai kelas

F1…Fn : Petunjuk atau syarat kondisi

3. IMPLEMENTASI

Dalam penyelesaian masalah seleksi pemilihan sistem praktikum di Filkom Universitas Brawijaya, data pada penelitian ini berupa atribut hasil *live* koding, mata kuliah, hasil mengajar, dan kelas. Kemudian hasil klasifikasi yang didapatkan adalah sangat disarankan, disarankan, dan yang terakhir tidak disarankan.

Implementasi berisi kode program dan tahapan-tahapan yang dilakukan pada *hadoop.* Implementasi pada permasalahan seleksi pemelihan asisten praktikum menggunakan bahasa pemrograman java. Dalam pengimplementasian menggunakan tiga class, satu class sebagai map, satu class lain sebagai reduce, dan class lainnya sebagai class main.

3.1 Kode Program

Berikut kode program dari kedua class, yaitu class Map.java dan class Reduce.java.

1. Map.java

```
public void map(LongWritable key, Text value,
Context context) throws IOException,
InterruptedException {
    if(test_input==null)
        test_input=context.getConfiguration().ge
        t("test_input").split("\\,");
        String[] input=value.toString().split("\\,");
        for(int j=0;j<input.length;j++){
            if(j==input.length-1){
```

```
        if(outcome_count.containsKey(input[j]))
                outcome_count.put(input[j],
        outcome_count.get(input[j])+1);
            else
                outcome_count.put(input[j],
        (double) 1);
        }
           else{

        if(input[j].contentEquals(test_input[j])){

                if(!inputs.containsKey(j+","+inpu
        t[j]))

                inputs.put(j+","+input[j], 0);

                if(features_count.containsKey(j+
        ","+input[j]+"|"+input[input.length-1]))

                features_count.put(j+","+input[j
        ]+"|"+input[input.length-1],
        features_count.get(j+","+input[j]+"|"+inp
        ut[input.length-1])+1);
            else
                features_count.put(j+","+input[j
        ]+"|"+input[input.length-1], (double) 1);
                            }
                }
        }
        ++count;
}
```

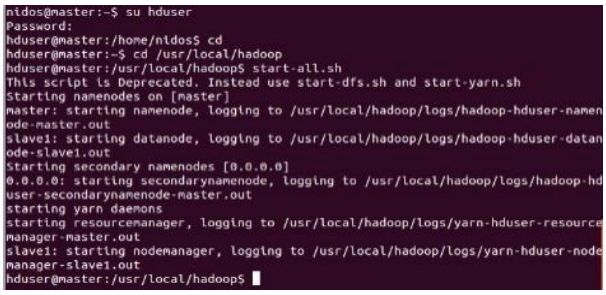
2. Reduce.java

```
public class NBCReduce extends
Reducer<IntWritable, Text, IntWritable, Text>{
public void reduce(IntWritable key,
Iterable<Text> values, Context context) throws
IOException, InterruptedException{
        Double
        out_value=Double.NEGATIVE_INFINIT
        Y;
        String out_key=null;
        HashMap<String,Integer>
        final_output=new
        HashMap<String,Integer>();
        for(Text value:values){
                if(final_output.containsKey(v
        alue.toString()))
                final_output.put(value.toStri
        ng(),
        final_output.get(value.toString())+1);
        else
                final_output.put(value.toStri
        ng(), 1);

}
        for(Entry<String,Integer>
        output:final_output.entrySet()){
        if(output.getValue()>out_value){

        out_value=(double)
output.getValue();
        out_key=output.getKey();
        }

}
        context.write(null, new
        Text(out_key));
        }

}
```

3.2 Langkah-langkah pada Hadoop

Langkah-langkah dimulai dari start sampai mengeluakan output dari metode *naïve bayes*:

1. Ketik start-all.sh

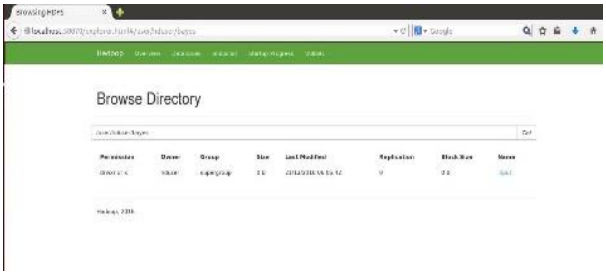


Gambar 3. Start-all.ssh

2. Membuat folder *bayes* di hduser, kemudian membuat folder *input* didalam folder *bayes*

Gambar 4. Membuat Folder

3. Mengecek folder di localhost:50070



Gambar 5. Cek Folder

4. *Compile* semua file java

```
hduser@master:/usr/local/hadoop$ bin/hdfs com.sun.tools.javac.Main NBC*.java
Note: NBCDriver.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
hduser@master:/usr/local/hadoop$
```

Gambar 6. *Compile* File Java

5. Mengecek pada direktori apakah file sudah menghasilkan *.class

Gambar 7. *Class* Pada Direktori

6. Membuat file *.jar dari hasil *.class

```
hduser@master:/usr/local/hadoop$ jar cf bayes.jar NBC*.class
hduser@master:/usr/local/hadoop$
```

Gambar 8. File *.jar

7. Mengecek pada direktori

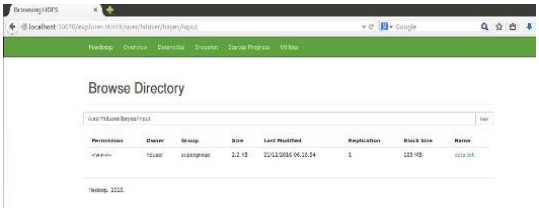
Gambar 9. Direktori

8. Memasukkan file data ke folder *input* localhost

```
hduser@master:/usr/local/hadoop$ bin/hdfs dfs -copyFromLocal /home/nidos/Downloa
ds/data.txt /user/hduser/bayes/input
hduser@master:/usr/local/hadoop$
```

Gambar 10. File Pada Folder *Input*

9. Mengecek apakah file data sudah di localhost:50070



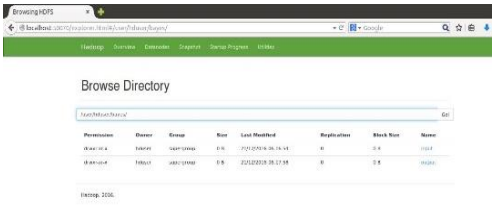
Gambar 11. File Pada *Localhost*

10. Membuat folder *output* di localhost:50070

```
hduser@master:/usr/local/hadoop$ bin/hdfs dfs -mkdir /user/hduser/bayes/output
hduser@master:/usr/local/hadoop$
```

Gambar 12. Folder *Output*

11. Mengecek folder *output* apakah sudah ada di localhost:50070



Gambar 13. Folder *Output*

12. Mengecek secara manual folder apa saja yang ada di localhost:50070

```
hduser@master:/usr/local/hadoop$ bin/hdfs dfs -ls /user/hduser/bayes
Found 2 items
drwxr-xr-x   - hduser supergroup          0 2016-12-21 06:16 /user/hduser/bayes/
input
drwxr-xr-x   - hduser supergroup          0 2016-12-21 06:17 /user/hduser/bayes/
output
hduser@master:/usr/local/hadoop$
```

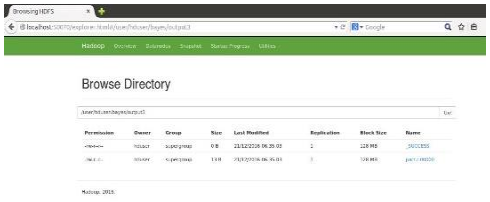
Gambar 14. Cek Folder

13. Proses menjalankan program

```
hduser@master:/usr/local/hadoop$ bin/hadoop jar bayes.jar NBCDriver lulus,rpl,ba
lk /user/hduser/bayes/input/data.txt /user/hduser/bayes/output3
16/12/21 06:34:54 INFO Configuration.deprecation: session.id is deprecated. Inst
ead, use dfs.metrics.session-id
16/12/21 06:34:54 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName
=JobTracker, sessionId=
16/12/21 06:34:55 WARN mapreduce.JobResourceUploader: Hadoop command-line option
 parsing not performed. Implement the Tool interface and execute your applicatio
n with ToolRunner to remedy this.
16/12/21 06:34:55 INFO input.FileInputFormat: Total input paths to process : 1
16/12/21 06:34:55 INFO mapreduce.JobSubmitter: number of splits:1
16/12/21 06:34:56 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_lo
cal310059613_0001
16/12/21 06:34:57 INFO mapreduce.Job: The url to track the job: http://localhost
:8080/
16/12/21 06:34:57 INFO mapreduce.Job: Running job: job_local310059613_0001
16/12/21 06:34:57 INFO mapred.LocalJobRunner: OutputCommitter set in config null
16/12/21 06:34:57 INFO output.FileOutputCommitter: File Output Committer Algorit
hm version is 1
16/12/21 06:34:57 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hado
op.mapreduce.lib.output.FileOutputCommitter
16/12/21 06:34:58 INFO mapred.LocalJobRunner: Waiting for map tasks
16/12/21 06:34:58 INFO mapred.LocalJobRunner: Starting task: attempt_local310059
613_0001_m_000000_0
16/12/21 06:34:58 INFO output.FileOutputCommitter: File Output Committer Algorit
hm version is 1
16/12/21 06:34:58 INFO mapred.Task:  Using ResourceCalculatorProcessTree : [ ]
16/12/21 06:34:58 INFO mapred.MapTask: Processing split: hdfs://master:9000/user
/hduser/bayes/input/data.txt:0+2202
16/12/21 06:34:58 INFO mapreduce.Job: Job job_local310059613_0001 running in ube
r mode : false
16/12/21 06:34:58 INFO mapreduce.Job:  map 0% reduce 0%
16/12/21 06:34:59 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
16/12/21 06:34:59 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
16/12/21 06:34:59 INFO mapred.MapTask: soft limit at 83886080
```

Gambar 15. Menjalankan Program

14. Melihat apakah hasil sudah keluar atau belum

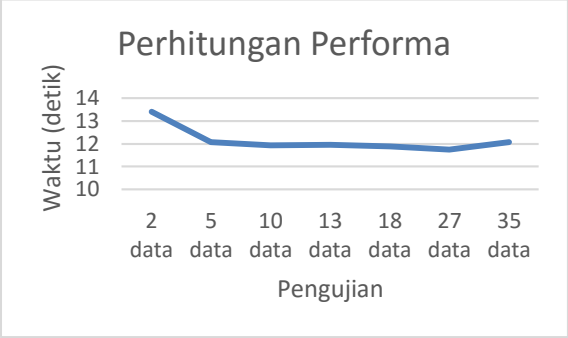


Gambar 16. Cek Hasil

4. PENGUJIAN DAN ANALISIS

Pengujian yang pertama adalah performa, hal ini dilakukan dengan menghitung waktu komputasi dari data. Dalam perhitungannya yaitu dihitung masing-masing data berapa lama waktu saat komputasi dijalankan. Pengujian ini dilakukan sebanyak 7 kali, kemudian dari 35 data tersebut dilakukan perhitungan rata-ratanya. Data yang digunakan diasumsikan akan terus disimpan setiap

kali ada perekrutan baru pada asisten praktikum sebagai data histori sekaligus data latih, sehingga seiring berjalannya waktu, maka data tersebut akan semakin besar atau masuk pada konteks *Big Data*. Pada Gambar 17 ditunjukkan bahwa waktu komputasi yang dibutuhkan sangat fluktuatif, karena memang datanya memang masih tergolong masih sedikit dan hanya sebagai simulasi dalam lingkungan hadoop multinode cluster. Selain itu, lama waktu juga tergantung dari aktifitas komputer.



Gambar 17. Grafik Pengujian Performa

Pengujian yang kedua adalah akurasi, hal ini dilakukan untuk melihat kualitas hasil data yang diuji terhadap ukuran seberapa besar kesesuaian dengan data aktual. Akurasi yang dihasilkan yaitu jumlah seluruh nilai yang benar dibagi dengan jumlah data. Hasil yang didapatkan 80 %.

5. KESIMPULAN DAN SARAN

Kesimpulan yang didapat adalah bahwa dengan menggunakan hadoop multinode dan metode *Naïve Bayes* yang digunakan untuk klasifikasi pemilihan asisten praktikum dapat diperoleh hasil yang cukup baik yaitu sebesar 80%. Saran yang dapat dilakukan dalam pengembangan selanjutnya yaitu, jumlah data yang digunakan dalam pengujian lebih banyak lagi dan adanya pengujian jumlah node pada PC slaves. Kemudian menerapkan metode lainnya yang ada pada *Big Data*. Dapat juga dilakukan *hybrid* atau penggabungan dengan 2 metode atau lebih agar hasil yang didapatkan lebih optimal.

6. DAFTAR PUSTAKA

(2013, Oktober). Diambil kembali dari Apache TM Hadoop @ homepage: http://hadoop.apache.org/.

BUSTAMI. 2013. Penerapan Algoritma Naive Bayes untuk Mengklasifikasi Data Nasabah Asuransi.

HAN, J. K. 2006. *Concept and Techniques* . Morgan Kaufmann.

INDUSTRI, M. 2013. *Definisi cloud Computing*. Diambil kembali dari Cloud Computing: Meruvian.org

IREM, D. 2016. Classifying Multi-Destination Trips in Austria with Big Data. *Elsevier*, 2211-9736.

KALAMSYAH, S. A. 2014. Aplikasi Pendukung Keputusan Seleksi Asisten Praktikum dan Lab Menggunakan Metode Analytical Hierarchi Process (Studi Kasus: Lab Informatika Universitas Telkom).

KUSUMADEWI, S. 2009. *Klasifikasi Status Gizi Menggunakan Naive Bayesian Classification.*

LUKITO, Y., & CHRISMANTO, A. R. 2015. Perbandingan Metode-Metode Klasifikasi Untuk Indoor Positioning System. *Jurnal Teknik Informatika dan Sistem Informasi*, 2.

SHOLIHAH, A. 2013. *Sistem Penentuan Uang Kuliah Tunggal (UKT) Menggunakan Metode Naive Bayes Classifier.*

SOEMARSO. 2003. *Akuntansi Suatu Pengantar*. Jakarta: Salemba Empat.

Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) p-ISSN: 2355-7699
Vol. 3, No. 4, Desember 2016, hlm. 279-284 e-ISSN: 2528-6579

SISTEM REKOMENDASI PEMILIHAN SEKOLAH MENENGAH ATAS SEDERAJAT KOTA MALANG MENGGUNAKAN METODE AHP ELECTRE DAN TOPSIS

Ibnu Aqli[1], Dian Eka Ratnawati[2], Mahendra Data[3]
[1,2,3]Fakultas Ilmu Komputer Universitas Brawijaya
Email: [1]ibnuaqli99@gmail.com, [2]dian_ilkom@ub.ac.id, [3]mahendra.data@ub.ac.id

(Naskah masuk: 1 Oktober 2016, diterima untuk diterbitkan: 26 Desember 2016)

Abstrak

Pemilihan tempat pendidikan yang bagus dan sesuai dengan kemampuan anak merupakan hal yang harus dikombinasikan untuk menunjang kemampuan perkembangan seorang anak. Apalagi pada masa pemilihan sekolah setelah lulus jenjang Sekolah Menengah Pertama (SMP) merupakan suatu keputusan yang harus dilakukan sambil mempertimbangkan masa depan. Dalam memilih sekolah lanjutan banyak hal yang biasanya dipertimbangkan, seperti Nilai Ujian Nasional (NUN) yang di dapat oleh siswa, jarak antar rumah siswa dan sekolah, fasilitas sekolah, bahkan prestasi-prestasi sekolah yang dianggap bisa menunjang kemampuan siswanya. Dari permasalahan tersebut, maka dirancang sebuah sistem untuk memberikan rekomendasi sekolah menengah atas sederajat di Kota Malang. Penelitian ini menerapkan metode *Analytical Hierarchy Process* (AHP) - *Elimination Et Choix Tranduisant La Realité* (ELECTRE) - *Technique for Order Preference by Similarity to Ideal Solution* (TOPSIS). AHP melakukan perhitungan pembobotan kriteria, ELECTRE melakukan klasifikasi alternatif "*favourable*", dan TOPSIS melakukan perankingan terhadap alternatif sehingga muncul rekomendasi sekolah yang sesuai dengan kriteria pengguna. Untuk pengujian, dilakukan uji akurasi pada metode TOPSIS dengan membandingkan data rekomendasi yang dikeluarkan oleh sistem dengan data yang didapat dari pakar. Pengujian akurasi pada metode TOPSIS mendapatkan nilai akurasi sebesar 82,98%.

Kata kunci: *Pendidikan, SMA, AHP, ELECTRE, TOPSIS*

Abstract

Selecting a good school and appropriate with the children's ability is a matter that must be combined to support children's development. Moreover, during the school admission time after junior high school period. This period is so essential that the parent and the child have to decide while considering the child's future. On deciding which Senior High Scool that will be attended, things that should be considered might be vary, such as the child's test score (NUN), the distance between home and the school, school facilities, and school achievements. From that issues, this paper explains a system to give a recommendation about Senior High Schools in Malang. This research applied Analytical Hierarchy Process (AHP) - Elimination Et Choix Tranduisant La Realite (ELECTRE) - Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) methods. AHP method is used to calculate criteria ELECTRE is used to classify alternative "favourable", and TOPSIS is used to make a rank through the alternatives so that appear several school recommendation that proper with user's criteria. To examine the system, the accurate test is conducted on TOPSIS method by comparing recommendation data issued by the system with the data issued by the expert. The accurate test on system get the value of 82.98% accurateness.

Keywords: *Education, Senior High School, AHP, ELECTRE, TOPSIS*

1. PENDAHULUAN

Pemilihan tempat pendidikan yang tepat merupakan hal yang harus dilakukan untuk menunjang kemampuan perkembangan seorang anak. Apalagi pada masa pemilihan Sekolah Menengah Atas (SMA) sederajat yang harus mempertimbangkan masa depan anak itu sendiri. Sekolah yang dapat dipilih oleh seorang anak adalah Sekolah Menengah Atas (SMA) atau Sekolah Menengah Kejuruan (SMK) ataupun yang lainnya. Dalam melakukan pemilihan sekolah lanjutan banyak hal yang biasanya dipertimbangkan, seperti Nilai Ujian Nasional (NUN) yang di dapat oleh siswa, jarak antar rumah siswa dan sekolah, fasilitas sekolah, bahkan prestasi-prestasi sekolah yang dianggap bisa menunjang kemampuan siswanya. Maka dari itu penulis memberikan sebuah solusi yaitu dengan membuat sistem rekomendasi pemilihan sekolah menengah atas sederajat Kota Malang menggunakan metode AHP ELECTRE TOPSIS.

AHP digunakan untuk melakukan pembobotan terhadap kriteria sehingga didapatkan pembobotan yang sesuai, ELECTRE digunakan untuk mengklasifikasikan alternatif yang ada berdasarkan kriteria dan pembobotan kritera yang didapatkan dari metode AHP. Sementara TOPSIS melakukan perankingan terhadap alternatif solusi yang diperoleh

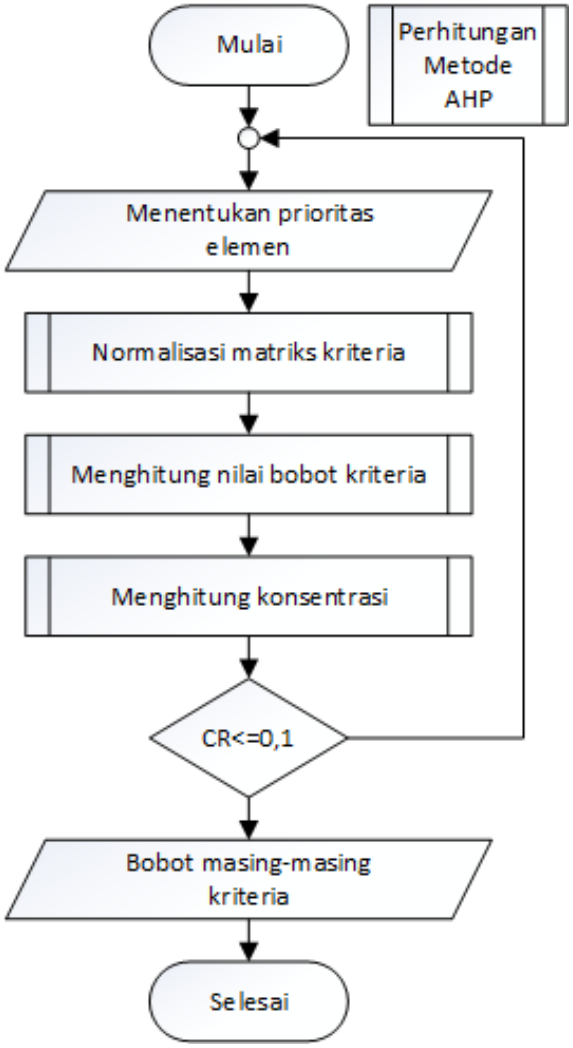
dari klasifikasi metode ELECTRE, hal ini dikarenakan ELECTRE tidak dapat melakukan proses perangkingan sehingga penggunaan metode TOPSIS diharapkan dapat menemukan alternatif yang terbaik. Diharapkan dengan menggabungkan metode tersebut dapat memberikan rekomendasi kepada orang tua wali murid ataupun pengguna lainnya dalam pemilihan Sekolah Menengah Atas (SMA) Sederjat di Kota Malang.

Penggunaan metode AHP Electre TOPSIS sebagai rekomendasi dari pemilihan telah dibahas penelitian sebelumnya, penelitian pertama yang dilakukan oleh Arinta Asesanti (2015) yang menggunakan metode ELECTRE dan TOPSIS pada Seleksi Penerimaan Peserta Didik Baru SMP Brawijaya Smart School (BSS) Kota Malang. Peda penelitiannya, asesanti menghasilkan akurasi sebesar 88,06%. Penelitian yang lain dilakukan oleh Jakti K. Prasojo (2016) yang menggunakan metode AHP dan TOPSIS untuk seleksi atlit pencak silat. Penelitian yang dilakukan oleh Jakti menghasilkan akurasi sebesar 83%. Penelitian yang dilakukan oleh Bramanti P. Pamungkas, (2016) yang menggunakan metode AHP dan ELECTRE untuk melakukan pemilihan pemain bola voli. Penelitian yang dilakukan oleh Bramanti menghasilkan akurasi sebesar 85,71%.

2. *ANALYTICAL HIERARCHY PROCESS* (AHP)

AHP adalah sebuah hierarki fungsional dengan input utamanya persepsi manusia. Model AHP memakai persepsi manusia yang dianggap "pakar" sebagai input utamanya. Kriteria "pakar" disini bukan berarti bahwa orang tersebut haruslah jenius, pintar, bergelar doktor dan sebagainya tetapi lebih mengacu pada orang yang mengerti benar permasalahan yang diajukan, merasakan akibat suatu masalah atau punya kepentingan terhadap masalah tersebut. (Suryadi, et al., 1998).

Flowchart jalannya metode AHP dapat dilihat pada Gambar 1. Pada awal proses AHP hal yang dilakukan adalah memasukkan prioritas elemen, prioritas elemen didapat dari inputan pengguna dengan mengurutkan 5 kriteria yang digunakan yaitu kriteria nilai, jarak antara sekolah dan tempat tinggal, prestasi sekolah, ekstrakulikuler yang ada di sekolah dan fasilitas yang dimiliki oleh sekolah. 5 kriteria tersebut diurutkan oleh pengguna dari yang paling penting ke prioritas yang kurang penting menurut pengguna. Kemudian, setelah prioritas elemen kriteria telah ditentukan, proses berikutnya adalah melakukan normalisasi matriks kriteria, lalu menghitung nilai bobot kriteria, dan menghitung konsentrasi. Setelah itu muncul nilai CR, apabila nilai CR lebih besar dari 0,1 maka proses dilakukan lagi mulai dari awal, yakni menentukan prioritas elemen kriteria. Tetapi apabila nilai CR kurang dari atau sama dengan 0,1, maka nilai bobot masing-masing kriteria dapat digunakan untuk proses berikutnya.



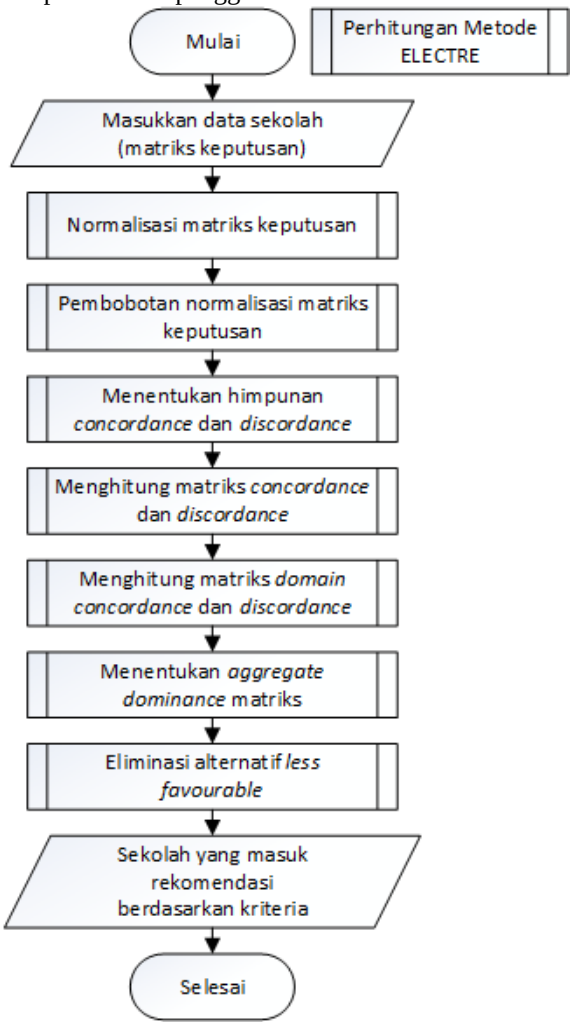
Gambar 1. Flowchart jalannya metode AHP

3. *ELIMINATION ET CHOIX TRANDUISANT LA REALITE* (ELECTRE)

Electre adalah singkatan dari *Elimination Et Choix Traduisant la Realite* atau dalam Bahasa inggris berarti *Elimination and Choice Expressing Reality*. Menurut Janko dan Bernoider (2005:11), Electre merupakan salah satu metode pengambilan keputusan multikriteria berdasarkan pada konsep outranking dengan menggunakan perbandingan berpasangan dari alternarif-alternatif berdasarkan setiap kriteria yang sesuai.

Flowchart jalannya metode Electre dapat dilihat pada Gambar 2. Pada awal proses ELECTRE hal yang dilakukan adalah memasukkan seluruh data sekolah yang digunakan dan telah disesuaikan dengan kriteria yang telah dimasukkan oleh pengguna. Kemudian, setelah data sekolah telah dimasukkan, dilakukan proses perhitungan electre, yaitu mulai dari normalisasi matriks keputusan, pembobotan normalisasi matriks keputusan, menentukan himpunan *concordance* dan *discordance*, menghitung matriks *concordance* dan *discordance*, menghitung matriks *domain concordance* dan

discordance, menghitung *aggregate dominance* matriks, dan yang terkahir adalah melakukan eliminasi alternatif yang *less favourable*. Setelah proses ELECTRE telah dilakukan, hasil keluaran proses ELECTRE adalah sekolah yang masuk rekomendasi berdasarakan kriteria yang telah diinputkan oleh pengguna.

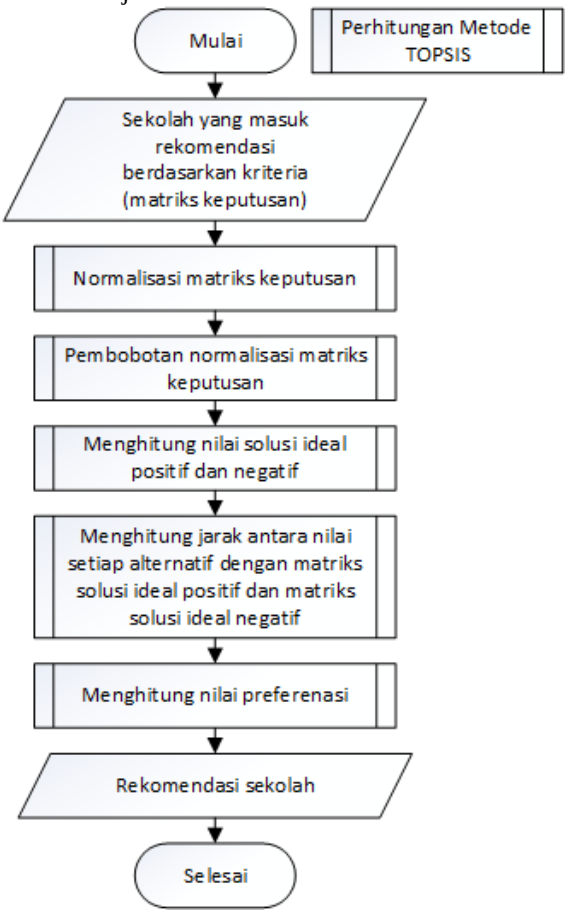


Gambar 2. Flowchart jalannya metode Electre

4. *TECHNIQUE FOR ORDER PREFERENCE BY SIMILARYTY TO IDEAL SOLUTION* (TOPSIS)

TOPSIS adalah akronim dari *Technique for Order Preference by Similarity of Ideal Solution.* TOPSIS merupakan salah satu metode pengambilan keputusan multikriteria yang pertama kali diperkenalkan oleh Yoon dan Hwang tahun 1981 (Juliyanti, et al., 2011). TOPSIS juga biasa digunakan untuk melakukan perangkingan dari beberapa alternatif yang ada.

Flowchart jalannya metode TOPSIS dapat dilihat pada Gambar 3. Pada awal proses TOPSIS, hal pertama yang dilakukan adalah memasukkan sekolah yang sesuai dengan kriteria yang telah dimasukkan oleh pengguna. Kemudian melakukan proses-proses yang terjadi dalam metode TOPSIS, yakni normalisasi matriks keputusan, pembobotan normalisasi matriks keputusan, menghitung nilai solusi ideal positif dan negatif, menghitung jarak antara nilai setiap alternatif dengan matriks solusi ideal positif dan matriks solusi ideal negatif, dan menghitung nilai preferensi. Setelah semua proses pada metode TOPSIS telah dilakukan, muncul 5 rekomendasi sekolah lanjutan yang dapat digunakan oleh pengguna sebagai pertimbangan memilih sekolah lanjutan.



Gambar 3. Flowchart jalannya metode TOPSIS

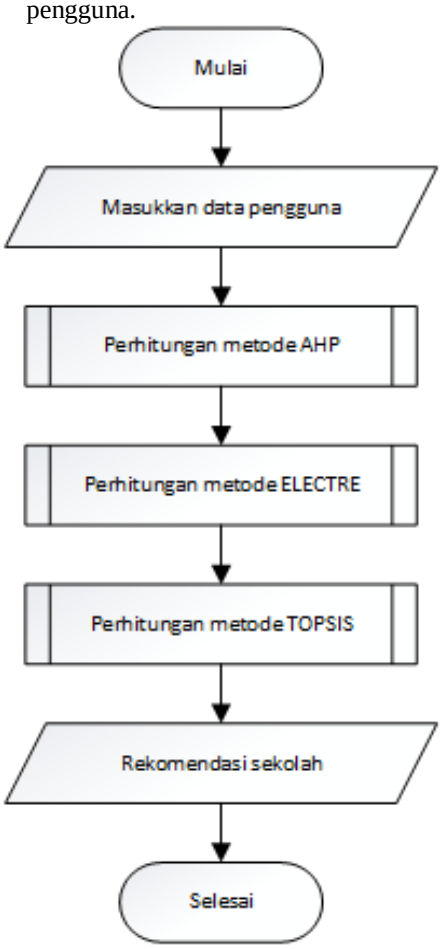
5. METODE

Implementasi sistem menjalankan 3 proses metode, yaitu metode AHP, metode Electre, dan metode TOPSIS. Flowchart jalannya sistem secara umum dapat dilihat pada Gambar 4.

Proses-proses yang terjadi pada sistem adalah sebagai berikut :

- Memasukkan data pengguna.
  Pertama yang dilakukan oleh sistem adalah menerima masukan data pengguna. Data pengguna yang dimasukkan adalah berupa :
  - Nama.
  - Alamat.
  - Pilihan lanjutan sekolah (SMA Negeri, SMA Swasta, SMK Negeri, SMK Swasta atau MA).
  - Jurusan (apabila memilih SMK Negeri atau SMK Swasta).

- Nilai (nilai ujian nasional dan nilai dari semester 1 sampai dengan semester 5 dari mata pelajaran matematika, bahasa indonesia, bahasa inggris, ilmu pengetahuan alam dan ilmu pengetahuan sosial).
- Prestasi sekolah yang diinginkan.
- Ekstrakulikuler yang diinginkan.
- Fasilitas sekolah yang diinginkan.
- Pilihan prioritas yang diinginkan oleh pengguna.



Gambar 4. Flowchart proses secara umum

- Proses pada metode AHP

  Pada proses metode AHP adalah proses pemberian bobot tiap kriteria yang diberikan kepada pengguna. Kriteria-kriteria yang digunakan adalah jarak dari rumah ke sekolah, nilai, prestasi sebuah sekolah, ekstrakulikuler yang dimiliki sekolah, dan fasilitan yang dimiliki sekolah. Pembobotan diisikan langung oleh pengguna, karena pengguna memiliki bobot tersendiri mengenai kriteria-kriteria tersebut.
- Proses pada metode Electre

  Pada proses metode electre adalah proses pengelompokan data. Data yang dikelompokkan adalah data pilihan sekolah. Data dikelompokkan sesuai kriteria yang telah dimasukkan oleh pengguna.
- Proses pada metode TOPSIS

  Pada proses metode topsis adalah proses perangkingan sekolah-sekolah yang telah dikelompokkan dan sesuai dengan kriteria dari pengguna. Dan hasil dari perangkingan tersebut adalah berupa rekomendasi sekolah yang direkomendasikan untuk pengguna sesuai dengan kriteria yang telah diinputkan.

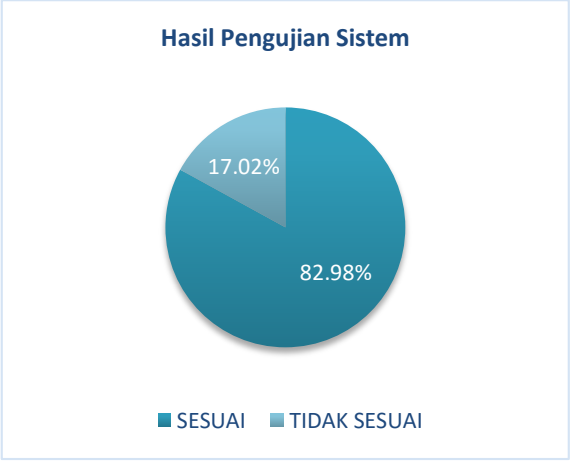
6. PENGUJIAN DAN ANALISIS

Pengujian yang dilakukan pada penelitian ini adalah pengujian akurasi. Pengujian dilakukan pada tahap hasil perankingan metode TOPSIS. Pengujian akurasi digunakan untuk mengetahui tingkat kesesuaian hasil keluaran sistem dengan data hasil survey yang didapat oleh penulis. Data uji yang digunakan adalah sebanyak 47 data.

Pengujian dilakukan dengan membandingkan data rekomendasi yang dikeluarkan oleh sistem dengan data yang didapatkan oleh penulis. Apabila sekolah yang ditempati oleh pengguna muncul pada halaman rekomendasi sekolah dan menempati posisi 5 besar, maka rekomendasi sistem bisa dikatakan sesuai.

Setelah pengujian dilakukan, diperoleh data yang sesuai sebesar 39, sedangkan data yang tidak sesui sebesar 8. Berdasarkan data tersebut akurasi sistem dapat diperoleh dengan perhitungan berikut :

$$Akurasi : \frac{(47-8)}{47} x\ 100\% = 82{,}98\%$$



Gambar 5. Hasil Pengujian Akurasi

Berdasarkan hasil pengujian, nilai akurasi yang didapat sebesar 82,98%. Pada pengujian hasil akurasi tidak mencapai 100% karena sistem dalam melakukan pengelompokan yang menggunakan metode electre hanya mendapatkan akurasi sebesar 85,11%, dan juga

hasil akurasi tidak mencapai 100% karena sistem dalam memberikan rekomendasi mempertimbangkan fasilitas yang dimiliki oleh sekolah, prestasi yang dimiliki oleh sekolah, dan ekstrakulikuler yang dimiliki oleh sekolah. Sedangkan dalam kenyataannya tiap sekolah memiliki fasilitas, prestasi dan ekstrakulikuler yang berbeda-beda.

7. KESIMPULAN

Berdasarkan hasil perancangan, implementasi, dan pengujian yang telah dilakukan pada bab sebelumnya, maka dalam penelitian ini dapat diambil kesimpulan :

1. Sistem rekomendasi dibangun dengan menggunakan metode AHP ELECTRE TOPSIS. Metode AHP digunakan untuk memberikan bobot pada tiap kriteria. Metode electre digunakan untuk mengkelompokkan sekolah yang sesuai dan tidak sesuai dengan kriteria yang telah dimasukkan oleh pengguna. Metode TOPSIS digunakan untuk melakukan perangkingan sekolah-sekolah yang sesuai dengan dengan kriteria yang dimasukkan oleh pengguna.
2. Sistem rekomendasi pemilihan sekolah menengah atas sederajat kota malang menggunakan metode AHP ELECTRE TOPSIS setelah dilakukan pengujian, ternyata memiliki akurasi sebesar 82,98%. Akurasi yang dimiliki sistem tidak dapat menacapai lebih dari 82,98% karena pada metode electre yang dilakukan pengelompokan data, juga memiliki akurasi yang tidak lebih dari 82,98%. Hal ini dapat terjadi karena sistem dalam memberikan rekomendasi mempertimbangkan fasilitas yang dimiliki oleh sekolah, prestasi yang dimiliki oleh sekolah, dan ekstrakulikuler yang dimiliki oleh sekolah. Sedangkan dalam kenyatannya tiap sekolah memiliki fasilitas, ektrakulikuler, dan prestasi yang berbeda-beda.

8. DAFTAR PUSTAKA

ABDULLAH, I., 2011. Sosiologi Pendidikan (Individu, Masyarakat, dan Pendidikan). Jakarta: PT Raja Grafindo Persada. Tersedia di <http://www.pengertianpakar.com/2015/03/sekolah-apa-itu-sekolah_7.html> [diakses 3 februari 2016]

AKHSAREARI. S., 2013. Sistem Pendukung Keputusan Pemilihan Produksi Sepatu Dengan Metode Elimination Et Choix Traduisant La Realite (Electre). S1. Universitas Pendidikan Indonesia

ASESANTI, ARINTA. 2015. "Sistem Pendukung Keputusan Seleksi Penerimaan Peserta Didik Baru SMP Menggunakan Metode ELECTRE-TOPSIS (Studi Kasus : SMP Brawijaya Smart School (BSS) Kota Malang)". PTIIK Universitas Brawijaya. Malang, Indonesia.

DINAS PENDIDIKAN, 2015. Panduang PPDB Penerimaan Peserta Didik Kota Malang Tahun Pelajaran 2015/2016.[pdf] Dinas Pendidikan. Tersedia di <https://statik.siap-ppdb.com/malang/content/unduhan/Panduan_PPDB_Online_KotaMalang_2015.00710066.pdf> [Diakses 2 Februari 2016]

FADLIL, J & MAHMUDY, WF 2007, 'Pembuatan sistem rekomendasi menggunakan decision tree dan clustering', Kursor, vol. 3, no. 1, pp. 45-66.

JULIYANTI, IRAWAN M.I DAN MUKLASH I. 2011. "Pemilihan Guru Berprestasi Menggunakan Metode AHP dan TOPSIS". Prosding Seminar Nasional Penelitian, Pendidikan dan Penerapan MIPA Universitas Negeri Yogyakarta. Yogyakarta

KUSRINI. 2007. "Konsep dan Aplikasi Sistem Pendukung Keputusan". Andi Offset, Yogyakarta.

MALANGPOST. 2015. Fantastis, Pergeseran Nilai PPDB, [online] tersedia di : <http://malang-post.com/pendidikan/88692-fantastis-pergeseran-nilai-ppdb> [Diakses 2 Februari 2016]

PAMUNGKAS, BRAMANTI PERMONO. 2016. "Sistem Pendukung Keputusan Pemilihan Pemain Bola Voli Menggunakan Metode AHP dan ELECTRE". PTIIK Universitas Brawijaya. Malang, Indonesia.

PPDB. 2015. Selamat Datang di SIAP PPDB Online, [online] tersedia di : <https://siap-ppdb.com/> [Diakses 3 Februari 2016]

PRASOJO, KINAYUNG JAKTI., REKYAN REGASARI, DAN SUTRISNO. 2015. "Implementasi Analytical Hierarchy Process – Technique For Order Preference By Similarity To Ideal Solution (AHP-TOPSIS) Untuk Penentuan Seleksi Atlet Pencak Silat". PTIIK Universitas Brawijaya. Malang, Indonesia.

SURYADI, KADARSAH DAN RAHMADHANI. 1998. Sistem pendukung keputusan. Bandung : PT remaja Rosdakarya

SAATY, T.L. DAN VARGAS, L.G. 2006, Decision making With The Analytic Network Process, sprinter. United Of America.

SCAFER, J.B.; KONSTAN, J.A. DAN RIEDL, J. 2001. Item-Based Collaborative Filtering Recommender Algorithms. WWW10.

UNDANG-UNDANG SISTEM PENDIDIKAN NASIONAL nomor 20 tahun 2003 tentang Sistem Pendidikan Nasional. Jakarta: Kementrian Sekretariat Negara Republik Indonesia.

Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) p-ISSN: 2355-7699
Vol. 3, No. 4, Desember 2016, hlm. 285-291 e-ISSN: 2528-6579

PREDIKSI NILAI TUKAR RUPIAH TERHADAP US DOLLAR MENGGUNAKAN METODE GENETIC PROGRAMMING

Daneswara Jauhari[1], Anang Hanafi[2], M. Fahrul Alam Y.[3], Arrofi Reza Satria[4], Luqman Hakim H.[5], Imam Cholissodin[6]

[1,2,3,4,5,6]Fakultas Ilmu Komputer Universitas Brawijaya
Email: [1]daneswarajauhari@gmail.com, [2]ananghanafi13@gmail.com, [3]fahrul.school@gmail.com, [4]arrofirezasatria@gmail.com, [5]hakim.harum@gmail.com, [6]imamcs@ub.ac.id

(Naskah masuk: 1 Oktober 2016, diterima untuk diterbitkan: 26 Desember 2016)

Abstrak

Nilai tukar mata uang mempunyai pengaruh yang sangat luas dalam perekonomian suatu negara, baik dalam negeri ataupun internasional. Pentingnya mengetahui pola nilai tukar *IDR* terhadap *USD* bisa membantu pertumbuhan ekonomi dikarenakan perdagangan luar negeri menggunakan mata uang negara yang berbeda. Maka dari itu, diperlukan suatu aplikasi yang dapat memprediksi nilai tukar Rupiah terhadap *US Dollar* di masa yang akan datang. Pada penelitian ini penulis menggunakan metode *genetic programming(GP)*, yang dapat menghasilkan solusi (*chromosome*) optimum, yang didapat dari evaluasi nilai tukar yang lalu, sehingga solusi ini digunakan sebagai pendekatan atau prediksi terhadap kurs nilai tukar mata uang Rupiah di masa yang akan datang. Solusi ini dibentuk dari kombinasi dari himpunan terminal (*set of terminals*) dan himpunan fungsi (*set of function*) yang dibangkitkan secara *random*. Setelah dilakukan pengujian dengan jumlah *popsize* dan iterasi yang berbeda, didapatkan bahwa Algoritma *GP* dapat melakukan prediksi nilai tukar Rupiah terhadap mata uang *US Dollar* dengan sangat baik, dilihat dari nilai *Mean Absolute Percentage Error (MAPE)* yang dihasilkan sebesar 0,08%. Penelitian ini bisa dikembangkan lebih baik dengan menambahkan parameter terminal dan parameter operasi sehingga bisa menambah variasi hasil perhitungannya.

Kata kunci: *prediksi, nilai tukar mata uang, genetic programming, MAPE*.

Abstract

Exchange currency rate has a wide influence in the economy of a country, both domestically or internationally. The importance of knowing the pattern of exchange rate against the IDR to USD could help the economic growth due to foreign trade involves the use of currencies of different countries. Therefore, we need an application that can predict the value of IDR against the USD in the future. In this research, the authors use genetic programming (GP) method which produces solutions (chromosome) that obtained from the evaluation of exchange rate and then this solution used as an approximation or prediction of currency exchange rate in the future. These solutions formed from the combination of the set terminal and the set of function that generated randomly. After testing by the number popsize and different iterations, it was found that the GP algorithm can predict the value of the rupiah against the US Dollar with a very good, judging from the value of Mean Absolute Percentage Error (MAPE) generated by 0.08%. This research can be developed even better by adding terminal parameters and operating parameters so they can add variation calculation results.

Keywords: *prediction, exchange currency rate, genetic programming, MAPE*.

1. PENDAHULUAN

Dua tahun yang lalu nilai tukar Rupiah terhadap *US Dollar* tidak stabil. Sebab-sebab yang mempengaruhi tidak stabil nya nilai tukar uang yaitu, tingkat inflasi pada negara, suku bunga pada negara, neraca perdagangan antar negara, jumlah permintaan barang ekspor, jumlah impor barang, hutang publik negara, kestabilan politik dan ekonomi negara (Anonymouse, 2015).

Ketidakstabilan nilai tukar uang, dapat membuat investor mengurungkan keinginannya untuk melakukan investasi, hal ini akan menyebabkan kemunduran pembangunan di Indonesia karena selama ini peran dari investor asing sangat besar dalam pertumbuhan ekonomi (Cecilya dkk., 2009). Hal ini akan semakin membuat nilai tukar *US Dollar* terhadap Rupiah semakin tinggi, karena sebagian besar transaksi internasional di Indonesia menggunakan *US Dollar*. Sementara itu nilai tukar rupiah akan menentukan indeks harga saham gabungan di Bursa Efek Jakarta (Kho dkk., 2011).

Oleh karena itu perlu adanya suatu aplikasi yang dapat memprediksi nilai tukar Rupiah terhadap *US Dollar* di masa yang akan datang.

Data historis akan dipelajari polanya untuk membuat sebuah prediksi. Metode yang dapat digunakan untuk melakukan prediksi ada banyak, seiring perkembangan teknologi dan informasi, metode untuk melakukan prediksi juga akan semakin berkembang.

Untuk mendapatkan hasil yang akurat dan stabil diperlukan sebuah metode yang tepat. Metode yang ada pada *Artifical Neural Networks*, dapat digunakan untuk mempelajari pola dari data historis yang didapatkan, *Artifical Neural Networks* meniru proses pembelajaran manusia, sehingga tepat untuk digunakan pada kasus sistem *non-linear*.

Penelitian mengenai prediksi dengan menggunakan metode yang ada pada *Artifical Neural Networks* pernah dilakukan oleh Jauhari (2016). Dalam penelitian ini di gunakan metode *Backpropagation Artifical Neural Networks*, penelitian tersebut bertujuan untuk memprediksi distribusi air PDAM Kota Malang, dan didapatkan hasil akurasi terbaik sebesar 97,99%.

Akan tetapi data nilai tukar uang merupakan data *non-linear* dengan banyak *noi*se. Hal ini akan mempersulit pengenalan pola yang dilakukan metode yang ada di *Artifical Neural Networks* (Kho dkk., 2011). Peneliti belum menemukan penelitian lain yang menggunakan metode *Genetic Programming*. Metode *Genetic Programming* merupakan satu metode secara matematis dan otomatis mengevolusi program komputer, berdasarkan sebuah permasalahan yang diberikan, metode ini dapat direpresentasikan dalam bentuk struktur pohon/*tree* (Mahmudy, 2016).

Genetic Programming telah diaplikasikan dalam berbagai bidang, *Genetic Programming* juga pernah digunakan juga untuk melakukan prediksi pada *respon time* sebuah layanan website, penelitian tersebut dilakukan oleh Fanjiang (2016), dan didapatkan hasil yang tidak kalah dengan algoritma pada *Artifical Neural Networks* yang biasa digunakan untuk memprediksi. Pada penelitian tersebut hasil yang didapatkan memiliki nilai rata-rata terbaik dibandingkan metode lain, artinya metode *Genetic Programming* dalam memprediksi mendapatkan hasil prediksi yang baik dan stabil.

Oleh karena itu untuk memprediksi nilai tukar Rupiah terhadap *US Dollar* pada penelitian ini diusulkan menggunakan metode *Genetic Programming*, yang diharapkan mendapatkan hasil prediksi yang baik dan stabil.

2. DASAR TEORI

2.1 Penjelasan Dataset

Data yang digunakan dalam penelitian adalah data nilai tukar mata uang Rupiah terhadap *Dollar* dari website fixer.io dari tahun 2000 sampai 2017 dengan mengambilnya melalui *api server* fixer.io dengan *JSON*. Dataset ini terdapat 5 kolom dengan variabel parameter X1, X2, X3, X4 dan Y. Tabel 2.1 ini berisi penjelasan tiap variabel pada dataset. Dan sampel dari dataset digambarkan pada Tabel 2.2, yang merupakan dataset yang diambil pada tanggal 29/12/2016 dengan jumlah 5.

Tabel 2.1 Keterangan Variabel

No	Kode Atribut	Keterangan
1	x1	Nilai aktual Rupiah pada H-4
2	x2	Nilai aktual Rupiah pada H-3
3	x3	Nilai aktual Rupiah pada H-2
4	x4	Nilai aktual Rupiah pada H-1
5	y	Nilai aktual pada hari H

Tabel 2.2 Dataset dari Fixer.io

x1	x2	x3	x4	y
13473	13435	13435	13435	13435
13435	13473	13435	13435	13435
13469	13435	13473	13435	13435
13398	13469	13435	13473	13435
13390	13398	13469	13435	13473

2.2 Nilai Tukar

Nilai tukar atau kurs merupakan suatu perbandingan antara nilai mata uang suatu negara dengan negara lain (Ardra, 2016). Selain itu, nilai Tukar adalah pertukaran antara dua mata uang yang berbeda, yaitu merupakan perbandingan nilai antara kedua mata uang tersebut (Triyono, 2008).

2.3 *Genetic Programming*

Genetic programming (GP) merupakan model dari pemrograman yang menggunakan ide-ide (dan beberapa terminologi) dari evolusi biologis untuk menangani masalah yang kompleks. Dari sejumlah program yang mungkin (biasanya fungsi program kecil dalam aplikasi yang lebih besar), program yang paling efektif akan bertahan dan bersaing atau *cross-breed* dengan program lain untuk terus lebih dekat dengan pendekatan solusi yang dibutuhkan.

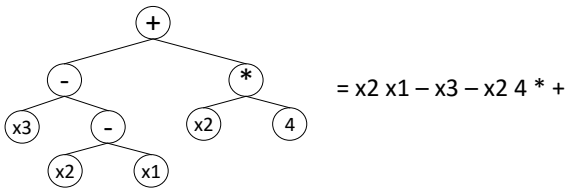
GP merupakan suatu model pendekatan yang terlihat paling sesuai dengan masalah-masalah yang memiliki variable-variable difluktuasi dalam jumlah besar. Berbeda dengan *Genetic Algorithm*, *Genetic Programming* menghasilkan program komputer yang direpresentasikan dalam struktur pohon (*tree*). (Mahmudy dkk., 2016)

GP bisa menggunakan proses rekombinasi dan mutasi dengan probabilitas tertentu, tapi tidak dapat keduanya. Rekombinasi dilakukan dengan cara saling menukar ranting (*sub tree*), sedangkan mutasi dilakukan dengan mungubah pohon secara acak. Sebuah bagian yang sulit dari penggunaan *GP* adalah menentukan fungsi fitness, sejauh mana program membantu untuk sampai ke tujuan yang diinginkan. Berbeda dengan pendekatan yang ada, pendekatan lain menggunakan teknik peramalan *time series* yang konvensional yang terbilang kurang akurat jika dibandingakan dengan *GP*.

Dua parameter yang merepresentasikan solusi dari *Genetic Programming* sebagai berikut:
- Penentuan himpunan terminal (*set of terminals*)
- Penentuan himpunan fungsi (*set of functions*)
(Mahmudy dkk., 2016)

2.3.1 Representasi Kromosom

Setiap kromosom yang mewakili calon solusi untuk memecahkan masalah merupakan struktur pohon yang cocok untuk operasi *GP*. Jika digunakan untuk menemukan program yang valid, maka setiap kromosom berevolusi dan dioperasikan oleh *GP* yang akan menjadi perwakilan program dalam bentuk pohon *(tree)*. Perhatikan representasi kromosom pada Gambar 2.1.



Gambar 2.1 Representasi Kromosom

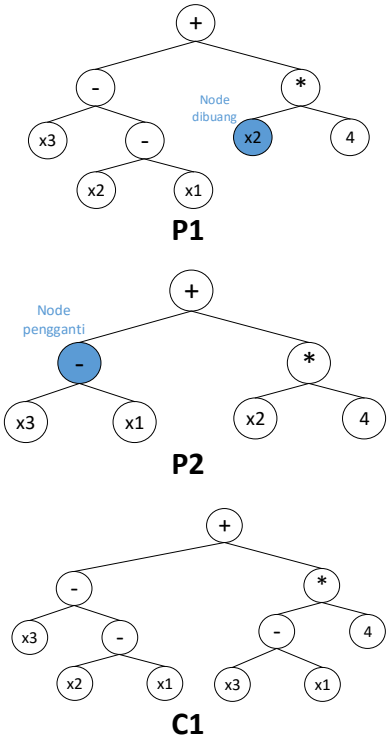
2.3.2 Inisialisasi dan Evaluasi

Sebelum menjalankan GP, operator harus menentukan elemen-elemen apa yang seharusnya ada dalam fungsi dan terminal set. Setelah itu GP mengambil beberapa *sample* atau *parents* secara *random* untuk diinisialisasi dan dievaluasi dengan menghitung nilai fitness dari individu tersebut.

Evaluasi individu dijalankan dengan menyusun fungsi non-linear dari binary tree. Sedangkan nilai fitness dihitung dari total selisih antara output fungsi *non-linear* dengan nilai y pada tabel.

2.3.3 Crossover

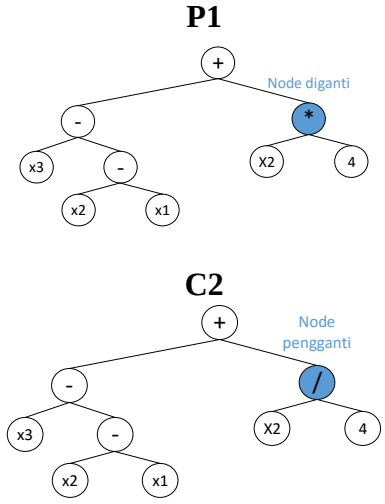
Operator menukar satu bagian (*gen*) dari satu induk kromosom dengan induk lainnya yang akan dicrossover. Setelah dua operasi seleksi selesai, kedua induk akan menghasilkan 2 *childs* yang akan dipilih dengan syarat memiliki kemampuan bertahan hidup yang *relative* lebih baik (bisa kita lihat dari nilai fitnessnya) dan akan diproses/digunakan untuk generasi berikutnya. Perhatikan proses crossover pada Gambar 2.2.



Gambar 2.2 Proses Crossover

2.3.4 Mutasi

Secara acak *GP* melakukan perubahan node dari sebuah kromosom menjadi node yang berbeda. Mutasi bisa meningkatkan keragaman evolusi pada suatu populasi dan meningkatkan eksplorasi *GP* dari ruang pencarian. Dimana, probabilitas mutasi biasanya ditetapkan jauh lebih rendah dari probabilitas crossover. Perhatikan proses mutasi pada Gambar 2.3.



Gambar 2.3 Proses Mutasi

2.3.5 Seleksi

Operator melakukan seleksi dengan mengambil kromosom dengan nilai fitness yang tertinggi, sehingga bisa bertahan hidup di generasi berikutnya. Proses dalam seleksi pada *GP* ini sama dengan *GAs*

yang biasanya menggunakan *binary tournament* dan *roulette-wheel*. Kromosom dengan fitness yang tertinggi, memiliki kesempatan yang besar untuk menjadi parent pada generasi berikutnya. Pada penelitian ini digunakan jenis seleksi *elitism* yang mengambil kromosom terbaik dari populasi.

2.4 Perhitungan Nilai Evaluasi

Untuk melakukan perhitungan evaluasi pada penelitian ini menggunakan persamaan *Mean Absolute Percentage Error (MAPE)* yang dihitung menggunakan kesalahan absolut pada tiap periode dibagi nilai observasi yang nyata. Setelah itu, dihitung rata-rata kesalahan persentase absolut. Pendekatan ini berguna ketika ukuran atau besar variabel ramalan itu penting dalam mengevaluasi ketepatan ramalan.

MAPE mencerminkan seberapa besar kesalahan dalam memprediksi yang dibandingkan dengan nilai nyata (Jauhari dkk., 2016). Rumus yang digunakan untuk menghitung *MAPE* terdapat pada persamaan 2.1 :

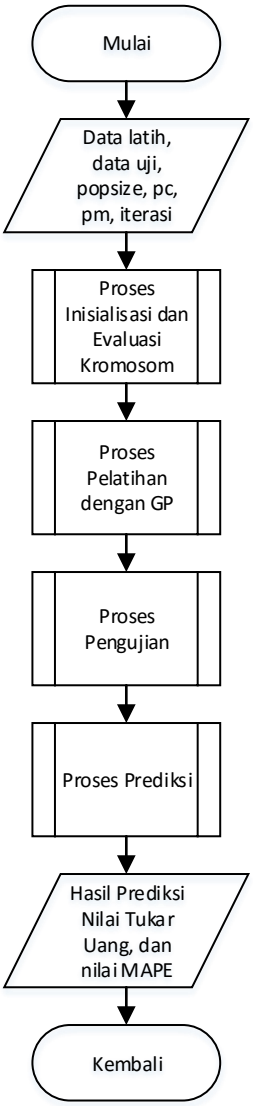
$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - yp_i|}{y_i} \times 100\% \qquad (2.1)$$

3. PERANCANGAN DAN IMPLEMENTASI

Proses dari prediksi nilai tukar uang dengan menggunakan metode *GP* melalui beberapa tahapan. Tahapan yang pertama yaitu melakukan proses inisialisasi dan evaluasi pada kromosom, kemudian tahap dua yaitu melakukan pelatihan dengan metode *GP*, pelatihan ini akan melewati beberapa proses yaitu:

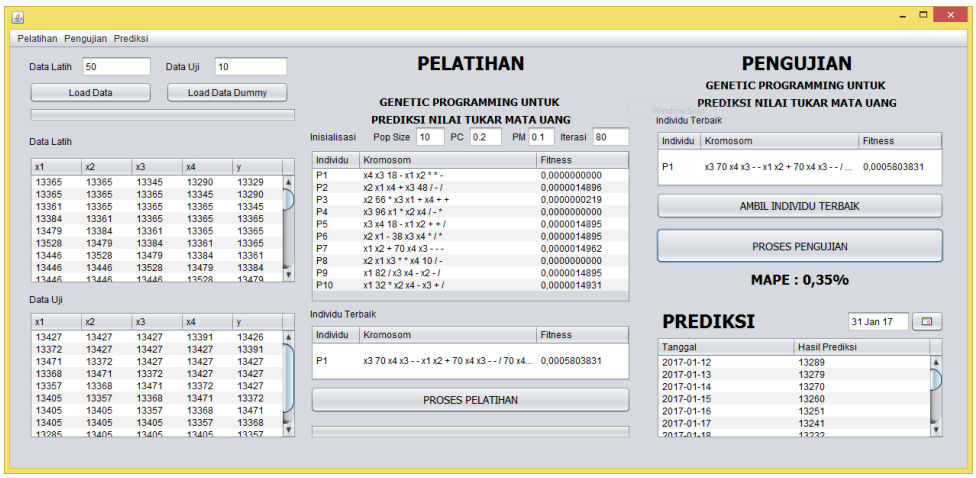
1. Crossover
2. Mutasi
3. Seleksi dengan metode *elitism*
4. Mendapatkan individu terbaik

Setelah dilakukan pelatihan, akan didapatkan individu terbaik, individu terbaik ini pada tahap ketiga digunakan untuk proses pengujian, tahap yang terakhir yaitu tahap 4 merupakan tahap prediksi nilai tukar uang. Tahapan-tahapan yang dilalui dapat digambarkan pada Gambar 3.1.



Gambar 3.1. *Flowchart* proses prediksi nilai tukar uang

Implementasi Antarmuka halaman utama, pada Gambar 3.2 merupakan tampilan halaman utama yang berisi input data, pelatihan, pengujian, dan prediksi.



Gambar 3.2. Antarmuka Halaman Utama

Implementasi Proses Postfix ini merupakan proses yang digunakan untuk membuat kromosom baru dengan cara random. Diawali dengan menentukan terminal dan operasi. Kemudian melakukan random sesuai dengan syarat postfix. Keseluruhan proses tersebut ditunjukkan dari potongan Kode Program 3.1.

```
1  public static String randomPostfix() {
2  String hasil = "";
3  Random rand = new Random();
4  // inisialisasi operator
5  String[] operator = {"+", "-", "*", "/"};
6  // inisialisasi terminal
7  String[] terminal = new String[5];
8  String[] tampungan = {"x1", "x2", "x3",
9  "x4", "0"};
10 tampungan[tampungan.length - 1] =
11 String.valueOf(rand.nextInt(100) + 1);
12
13 ArrayList<Integer> random = new
14 ArrayList<Integer>();
15 for (int r = 0; r < tampungan.length; r++)
16 {
17 random.add(new Integer(r));
18 }
19 Collections.shuffle(random);
20 for (int j = 0; j < tampungan.length; j++)
21 {
22 terminal[j] = tampungan[(int)
23 random.get(j)];
24 }
25
26 // buat postfix random
27 int A = 0;
28 int O = 0;
29 for (int i = 0; i < ((operator.length +
30 terminal.length)); i++) {
31 if (i < 2) {
32 hasil += terminal[A] + " ";
33 A++;
34 } else if (i == (operator.length +
35 terminal.length) - 1) {
36 hasil += operator[rand.nextInt(4)];
37 } else {
38 if (A >= terminal.length) {
39 hasil += operator[rand.nextInt(4)] + " ";
40 O++;
41 } else if (A == (O + 1)) {
42 hasil += terminal[A] + " ";
43 A++;
44 } else {
45 if ((rand.nextInt(2) + 1) % 2 == 1) {
46 hasil += operator[rand.nextInt(4)] + " ";
47 O++;
48 } else {
49 hasil += terminal[A] + " ";
50 A++;
51 }
52 }
53 }
54 }
55 return hasil;
56 }
```

Kode Program 3.1. Proses Random Postfix

Penjelasan dari Kode Program 3.1:

1. Baris 2-12 merupakan proses inisialisasi operator, terminal, dll.
2. Baris 13-24 merupakan proses pembentukan random untuk terminal.
3. Baris 27-28 merupakan proses inisialisasi angka dan operator.
4. Baris 29-30 melakukan perulangan sampai jumlah operator + jumlah terminal
5. Baris 31-33 jika jumlah i masih kurang dari 2 maka isi dengan angka
6. Baris 34-37 jika jumlah i merupakan urutan terakhir
7. Baris 38-41 jika jumlah A lebih besar dari panjang terminal maka isi dengan operator
8. Baris 42-44 jika jumlah O = A + 1 maka isi dengan terminal
9. Baris 45-48 jika nilai random ganjil maka isi operator
10. Baris 49-51 jika nilai random genap maka isi terminal
11. Baris 55 kembalikan nilai hasil

4. PENGUJIAN DAN ANALISIS

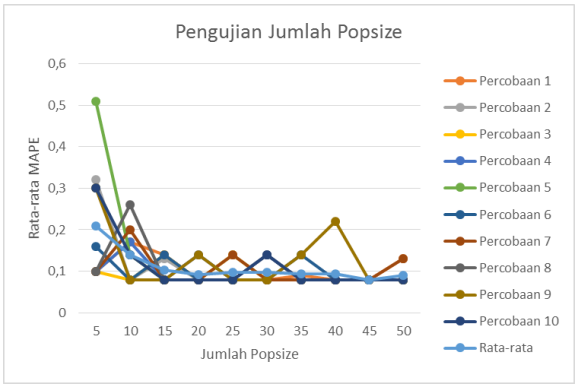
4.1 Pengujian Jumlah Popsize

Pengujian *popsize* digunakan untuk menentukan *popsize* yang optimal untuk mendapatkan nilai *MAPE* yang minimum. Parameter yang digunakan pada pengujian ini adalah jumlah data latih 50, jumlah data uji 10, jumlah *popsize* pada range 5-50, nilai pc 0,2, nilai pm, 0.1, dan jumlah iterasi 100. Pengujian jumlah *popsize* ini dilakukan sebanyak 10 kali. Hasil pengujian dapat dilihat pada Tabel 4.1 berikut ini:

Tabel 4.1 Pengujian Jumlah *Popsize*

Jumlah *Pop Size*	*MAPE* (%) Nilai Percobaan Ke-i										Rerata *MAPE* (%)
	1	2	3	4	5	6	7	8	9	10	
5	0,1	0,32	0,1	0,1	0,51	0,16	0,1	0,1	0,3	0,3	0,209
10	0,17	0,08	0,08	0,17	0,14	0,08	0,2	0,26	0,08	0,14	0,14
15	0,14	0,13	0,14	0,08	0,08	0,14	0,08	0,08	0,08	0,08	0,103
20	0,08	0,08	0,08	0,14	0,08	0,08	0,08	0,08	0,14	0,08	0,092
25	0,08	0,14	0,14	0,08	0,08	0,08	0,14	0,08	0,08	0,08	0,098
30	0,08	0,08	0,08	0,14	0,08	0,08	0,08	0,14	0,08	0,14	0,098
35	0,09	0,08	0,08	0,08	0,08	0,14	0,08	0,08	0,14	0,08	0,093
40	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,22	0,08	0,094
45	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08
50	0,13	0,08	0,08	0,08	0,08	0,08	0,13	0,08	0,08	0,08	0,09

Berdasarkan grafik hasil pengujian pada Gambar 4.1, rata-rata nilai *MAPE* terkecil didapatkan pada popsize 45 dengan nilai 0.08, yang artinya semakin banyak jumlah *popsize*, maka akan didapatkan nilai *MAPE* yang minimum.



Gambar 4.1. Hasil Pengujian Jumlah *Popsize*

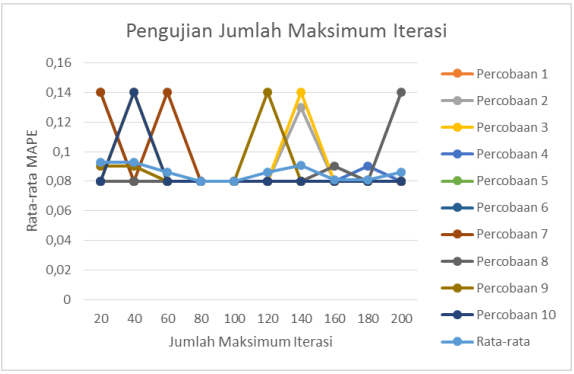
4.2 Pengujian Jumlah Maksimum Iterasi

Pengujian jumlah maksimum iterasi digunakan untuk menentukan jumlah maksimum iterasi yang optimal untuk mendapatkan nilai *MAPE* yang minimum. Tentunya dengan menggunakan jumlah *popsize* yang optimal pada pengujian sebelumnya. Parameter yang digunakan pada pengujian ini adalah jumlah data latih 50, jumlah data uji 10, jumlah *popsize* 45, nilai pc 0,2, nilai pm, 0.1, dan jumlah iterasi pada range 20-200. Pengujian jumlah maksimum iterasi ini dilakukan sebanyak 10 kali. Hasil pengujian dapat dilihat pada Tabel 4.2 berikut ini:

Tabel 4.1 Pengujian Jumlah *Popsize*

Jumlah Maksimum Iterasi	*MAPE* (%) Nilai Percobaan Ke-i										Rerata *MAPE* (%)
	1	2	3	4	5	6	7	8	9	10	
20	0,08	0,14	0,08	0,08	0,08	0,08	0,14	0,08	0,09	0,08	0,093
40	0,08	0,08	0,14	0,08	0,08	0,08	0,08	0,08	0,09	0,14	0,093
60	0,08	0,08	0,08	0,08	0,08	0,08	0,14	0,08	0,08	0,08	0,086
80	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08
100	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08
120	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,14	0,08	0,086
140	0,08	0,13	0,14	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,091
160	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,09	0,08	0,08	0,081
180	0,08	0,08	0,08	0,09	0,08	0,08	0,08	0,08	0,08	0,08	0,081
200	0,08	0,08	0,08	0,08	0,08	0,08	0,08	0,14	0,08	0,08	0,086

Berdasarkan grafik hasil pengujian pada Gambar 4.2, rata-rata nilai *MAPE* terkecil didapatkan pada jumlah maksimum iterasi sebesar 80 dan 100 dengan nilai *MAPE* 0,08. Ketika jumlah maksimum iterasi 80 dan 100 sudah didapatkan jumlah maksimum iterasi yang optimal, hal ini dapat simpulkan karena nilai *MAPE* tidak ada yang lebih kecil lagi di jumlah maksimum iterasi setelahnya.



Gambar 4.2. Hasil Pengujian Jumlah Maksimum Iterasi

5. KESIMPULAN DAN SARAN

Berdasarkan hasil uji coba parameter algoritma *Genetic Programming* untuk melakukan prediksi nilai tukar Rupiah terhadap *US Dollar*. Didapatkan bahwa Algoritma *GP* dapat melakukan prediksi nilai tukar Rupiah terhadap *US Dollar* dengan sangat baik, dengan parameter algoritma *GP* yang digunakan terdapat 4 operasi yakni "+", "-", "*", "/", "," dan juga 5 terminal yakni x1, x2, x3, x4, x, yang merupakan data nilai tukar uang Rupiah terhadap *US Dollar*.

Untuk mengukur solusi dari permasalahan prediksi nilai tukar Rupiah terhadap *US Dollar*, pada penelitian ini menggunakan perhitungan nilai *MAPE*. Penghitungan menggunakan parameter terbaik dapat menghasil nilai *MAPE* sebesar 0,08%. Parameter terbaik dengan rata-rata nilai *MAPE* terendah adalah sebagai berikut :

a. Jumlah data latih : 50
b. Jumlah data uji : 10
c. Jumlah popsize : 45
d. Nilai PC : 0.2
e. Nilai PM : 0.1
f. Jumlah Iterasi : 80 atau 100

Penelitian ini dapat dikembangkan dengan menambahkan jumlah parameter terminal, dan juga menambah variasi dari parameter operasi. Penggunaan nilai terminal yang lebih banyak akan mempengaruhi hasil variasi dari perhitungannya, penambahan variasi dari operasi juga akan mengatasi permasalahan ketika kromosom terdapat operasi pembagian. Selain itu bisa dicoba untuk menerapkan algoritma pengembangan dari *GP* yang telah ada saat ini, sehingga dapat mendapatkan hasil yang lebih baik.

6. DAFTAR PUSTAKA

ANONYMOUSE, 2015. Mengapa Mata Uang Dunia dan Nilai Kurs Negara Beda-beda. Tersedia di: http://uangindonesia.com/mengapa-mata-uang-dunia-dan-nilai-kurs-negara-beda/ [Diakses 05 Desember 2016].

ARDRA. 2016. Pengertian Nilai Tukar. Tersedia di: https://ardra.biz/ekonomi/valuta-asing/kurs-valuta-asing/ [Diakses 29 Desember 2016].

CECILYA, ARANTIKA. dkk., 2009. Prediksi Nilai Tukar US Dollar Terhadap Rupiah Menggunakan Algoritma Genetika Dan Elman Recurrent Neural Network, Universitas Telkom.

MAHMUDY, F.M. 2016. Modul Algoritma Evolusi Semester Ganjil 2016-2017, Universitas Brawijaya.

FANJIANG, YONG-YI. dkk. 2016. Search based approach to forecasting QoS attributes of web services using genetic programming. *Information and Software Technology 80 (2016) 158-174*.

JAUHARI, DANESWARA. dkk. 2016. Prediksi Distribusi Air PDAM Menggunakan Metode Jaringan Syaraf Tiruan Backpropagation di PDAM Kota Malang. *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK), 3(2), 85-89.*

KHO, DICKY KHOSMAN. dkk. 2011. Prediksi Nilai Tukar IDR / USD Menggunakan Artificial Neural Networks Dengan Metode Pembelajaran Ant Colony Optimization, Universitas Telkom.

TRIYONO. 2008. Analisis Perubahan Kurs Rupiah Terhadap Dollar Amerika. Jurnal Ekonomi Pembangunan. Vol.9 No. 2, Desember 2008: 156-167. Universitas Muhammadiyah Surakarta.

Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) p-ISSN: 2355-7699
Vol. 3, No. 4, Desember 2016, hlm. 292-299 e-ISSN: 2528-6579

REVIEW METODE PENDETEKSIAN GOD CLASS

Divi Galih Prasetyo Putri[1], Muhammad Shulhan Khairy[2], Siti Rochimah[3]

[1]Departemen Teknik Elektro dan Informatika, Sekolah Vokasi Universitas Gadjah Mada
[2,3]Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember
Email: [1]divi.galih@ugm.ac.id, [2]khairy10@mhs.if.its.ac.id, [3]siti@if.its.ac.id

(Naskah masuk: 1 Oktober 2016, diterima untuk diterbitkan: 26 Desember 2016)

Abstrak

Code Smell mengacu pada konsep mengenai pola atau aspek desain pada sistem perangkat lunak yang dapat menimbulkan masalah dalam proses pengembangan, penggunaan, atau perawatan sebagai dampak dari implementasi yang buruk dari desain perangkat lunak. Code Smell dapat menurunkan aspek *understandability* dan *maintainability* program. Program yang mengandung God Class juga cenderung lebih sulit untuk dirawat dibandingkan dengan program yang sama namun tidak mengandung God Class. God Class atau dapat juga disebut Blob merupakan sebuah kelas yang terlalu banyak berisi fungsionalitas didalamnya. Kelas-kelas seperti ini mengolah dan mengakses banyak informasi sehingga sulit dipahami. Pada penelitian ini akan dibahas metode-metode untuk mendeteksi adanya God Class. Selain itu juga dibandingkan kelebihan serta kekurangan metode-metode yang telah dianalisa. Dari pencarian literatur yang dilakukan, didapatkan 3 buah metode, metode pertama menggunakan cara deteksi dalam bentuk *rule card*, metode kedua menggunakan *rule card* dan catatan histori perubahan pada sebuah perangkat lunak, dan metode ketiga adalah pendeteksian berdasarkan contoh kelas yang dideteksi manual sebagai kecacatan perangkat lunak. Dari ketiga metode tersebut, metode ketiga dinilai sebagai yang terbaik berdasarkan nilai presisi dan *recall*-nya.

Kata kunci: *Blob, God Class*

Abstract

Code smell referring to the concept about a pattern or design aspects on a software system that can make a problem in the process of development, using, or maintenance as the impact of bad implementation of software design. Code smell can lower software understandability and maintainability. A software that containing god class will be more difficult to maintain compared with a same software but doesn't have a god class. God class, also called blob is a class that has too many functionality. A god class process and access a lot of information. On this research will be discussed methods to detect a god class. We also compared the advantage and disadvantage about analysed method. From the literature we search, there are 3 methods, first method using detection with a rule card, the second method using rule card and history changes of a software, and the third method is detection by examples classes that detected manually as a software defect. And our research result is the third method is the best method based on its precision and recall.

Keywords: *Blob, God Class*

1. LATAR BELAKANG

Code Smell mengacu pada konsep mengenai pola atau aspek desain pada sistem perangkat lunak yang dapat menimbulkan masalah dalam proses pengembangan, penggunaan, atau perawatan sebagai dampak dari implementasi yang buruk dari desain perangkat lunak. Pada penelitian terdahulu telah dikaji mengenai dampak Code Smell terhadap frekuensi perubahan pada kode program. Code Smell dapat menurunkan aspek *understandability* dan *maintainability* program. Jika sebuah perangkat lunak sulit untuk dimengerti dan dirawat oleh pengembang selanjutnya maupun tim perawat perangkat lunak, maka dapat merugikan pengembang sistem karena proses *maintenance* merupakan proses yang sangat penting dengan biaya yang cukup besar.

Dalam sebuah penelitian diketahui bahwa program yang mengandung God Class dan *Shotgun Surgery* lebih sering mengalami perubahan. Program yang mengandung God Class juga cenderung lebih sulit untuk di-*maintenance* dibandingkan dengan program yang sama namun tidak mengandung God Class. Pada (Sjøberg, et al., 2013) aspek *maintenance* terkena dampak terbesar jika terdapat Code Smell, termasuk God Class. God Class atau dapat juga disebut Blob merupakan sebuah kelas yang berisi terlalu banyak fungsionalitas didalamnya. Kelas-kelas seperti ini mengolah dan mengakses banyak informasi sehingga sulit dipahami. Padahal seharusnya sebuah kelas memiliki fungsi yang spesifik, sehingga konsep modularitas dalam pemrograman berorientasi objek

tetap terjaga dan tidak menimbulkan dampak sulitnya perawatan perangkat lunak kedepannya. God Class juga salah satu jenis Code Smell yang paling sering muncul dalam perangkat lunak. Jenis lainnya adalah *Duplicate Code*, *Data Class*, *Schizophrenic Class* dan *Long Method*, hal ini dibahas pada penelitian (Fontana, et al., 2013). Selain itu terdapat juga penelitian (de F. Carneiro, et al., 2010) yang melakukan visualisasi adanya Code Smell, termasuk God Class.

Dari referensi literatur terkait pendeteksian Code Smell, telah banyak dikembangkan metode untuk mendeteksi adanya Code Smell pada sebuah sistem. *Paper* ini memiliki kontribusi dalam pembahasan metode-metode untuk mendeteksi adanya God Class. Selain itu juga membandingkan kelebihan serta kekurangan metode-metode yang telah dianalisa. Dari kontribusi ini diharapkan akan terdapat studi-studi selanjutnya tentang pendeteksian God Class agar permasalahan tersebut dapat diminimalisir untuk pengembang perangkat lunak kedepannya.

2. PENELITIAN TERKAIT

Telah dibahas mengenai 22 tipe dari Code Smell dengan karakteristik yang berbeda dan efek yang berbeda pada sistem, hal ini dibahas pada (Fowler, 1999). Berbagai macam Code Smell tersebut mempengaruhi struktur dari kode dan membuka peluang untuk dilakukan *refactoring*. Salah satu jenis Code Smell adalah God Class yang mengacu pada kelas yang menjadi sentral dari sistem, dibahas pada penelitian (Lanza, et al., 2005). Penelitian tersebut juga mengajukan sebuah matriks untuk mendeteksi God Class.

Pada penelitian sebelumnya (Zhang & Hall, 2011) telah dibuat studi literatur mengenai perkembangan penelitian terhadap Code Smell dari tahun 2000-2009. Penelitian lainnya (Hamid & Ilyas, 2013) menyusun sebuah studi literatur terhadap kakas bantu pendeteksi Code Smell. Disini dibandingkan kemampuan dua buah kakas bantu dalam mendeteksi dua jenis Code Smell yaitu Feature Envy dan God Class.

Dari beberapa penelitian terkait tersebut, belum ada penelitian yang secara khusus melakukan studi terhadap metode-metode pendeteksian Code Smell terutama untuk jenis God Class. Oleh karena itu, diharapkan penelitian ini dapat memberikan kontribusi.

3. METODE PENELITIAN

3.1. Pertanyaan Penelitian

Berdasarkan studi literatur pada penelitian-penelitian sebelumnya, *paper* ini bertujuan untuk mempelajari metode pendeteksian God Class dan melakukan analisis terhadap hasilnya. Untuk itu dibentuklah *research questions* sebagai berikut:

RQ1: Metode-metode apa saja yang telah dikembangkan untuk mendeteksi adanya God Class?
RQ2: Bagaimana hasil dalam pendeteksian God Class pada setiap metode?
RQ3: Apa saja efek yang ditimbulkan oleh God Class terhadap sistem?

3.2. Prosedur Pencarian

Sebelum melakukan proses studi literatur perlu dilakukan proses pencarian literatur yang memiliki kesesuaian dengan topik dan *research question* yang telah ditentukan. Pencarian dilakukan pada sumber-sumber elektronik berikut:

- IEEExplore (http://ieeexplore.ieee.org), dan
- Screince Direct (http://sciencedirect.com).

Pencarian hanya dilakukan pada dua sumber karena kedua sumber tersebut adalah sumber jurnal maupun *paper* konferensi yang banyak dikenal dan menjadi sumber rujukan penelitian dimanapun, peneliti juga hanya memiliki akses pada dua sumber elektronik tersebut. Untuk pencarian pada repositori literatur Sciencedirect dilakukan pencarian hanya untuk literatur dengan akses terbuka, karena literatur dengan akses tertutup tidak dapat dibuka kecuali memiliki akses penuh pada sumber tersebut. Berbeda halnya dengan literatur IEEE Xplore yang dapat diakses secara penuh dengan koneksi internet dalam kampus. Pada proses pencarian literatur ditetapkan beberapa kata kunci pencarian:

- "Software Evolution"
  AND
  "Bad Smell Code"
- "Software Evolution"
  AND
  "Bad Smell Detection"
- "Software Evolution"
  AND
  "Code Smell Detection"
- "Bad Smell Detection"
  AND
  "God Class"
- "Code Smell Detection"
  AND
  "God Class"
- "Software Quality"
  AND
  "God Class"

3.3. Proses Seleksi Studi

Pada subbab ini akan dibahas tentang hasil pencarian kata kunci diatas pada dua sumber elektronik yang telah diterangkan sebelumnya.

Tabel 1. Hasil pencarian literatur dengan penyaringan kata kunci

Sumber Elektronik	Kata Kunci	Hasil Pencarian
IEEExplore	Software Evolution AND Bad Smell Code	12
Science Direct		4
	Total	16
IEEExplore	Software Evolution AND Bad Smell Detection	3
Science Direct		4
	Total	7
IEEExplore	Software Evolution AND Code Smell Detection	9
Science Direct		11
	Total	20
IEEExplore	Bad Smell Detection AND God Class	0
Science Direct		26
	Total	26
IEEExplore	Code Smell Detection AND God Class	3
Science Direct		25
	Total	28
IEEExplore	Software Quality AND God Class	7
Science Direct		6
	Total	13
	Total	110

Total hasil pencarian secara keseluruhan berjumlah 110 dokumen. Pencarian ini hanya dilakukan dengan penyaringan kata kunci. Kemudian dilakukan seleksi berdasarkan *inclusion* dan *exclusion criteria* sebagai berikut.

Tabel 2. Daftar *inclusion criteria* dan *exclusion criteria*

Inclusion Criteria	*Exclusion Criteria*
Literatur merupakan *paper* jurnal atau *paper conference*.	Literatur dengan topik yang sama.
Literatur diterbitkan pada rentang tahun 2010-2014	Literatur sebelum tahun 2010.
Literatur menggunakan bahasa inggris.	Literatur yang tidak berkaitan dengan God Class dan metode deteksinya.
Literatur dapat diakses secara *online*.	
Literatur yang berkaitan dengan God Class dan metode deteksinya.	

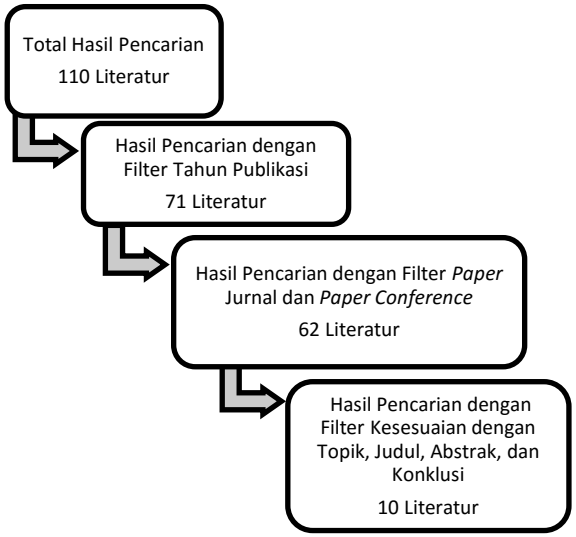
Tabel 3. Hasil pencarian literatur dengan penyaringan tambahan tahun publikasi

Sumber Elektronik	Kata Kunci	Tahun Publikasi	Hasil
IEEExplore	Software Evolution AND Bad Smell Code	2010-Sekarang	3
Science Direct			8
	Total		11
IEEExplore	Software Evolution AND Bad Smell Detection	2010-Sekarang	2
Science Direct			3
	Total		5
IEEExplore	Software Evolution AND Code Smell Detection	2010-Sekarang	5
Science Direct			6
	Total		11
IEEExplore	Bad Smell Detection AND God Class	2010-Sekarang	0
Science Direct			19
	Total		19
IEEExplore	Code Smell Detection AND God Class	2010-Sekarang	2
Science Direct			18
	Total		20
IEEExplore	Software Quality AND God Class	2010-Sekarang	1
Science Direct			4
	Total		5
		Total	71

Tabel 4. Hasil pencarian literatur dengan penyaringan tambahan jenis literatur

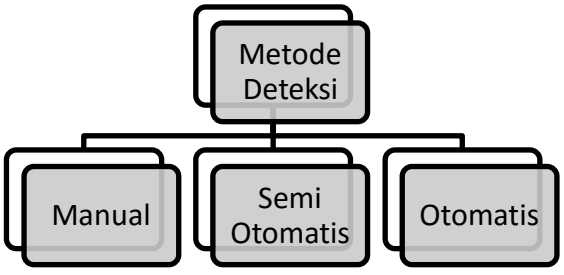
Sumber Elektronik	Kata Kunci	Tahun Publikasi	Filter *Paper*	Hasil
IEEExplore	Software Evolution AND Bad Smell Code	2010-Sekarang	Ya	8
Science Direct				3
	Total			11
IEEExplore	Software Evolution AND Bad Smell Detection	2010-Sekarang	Ya	2
Science Direct				3
	Total			5
IEEExplore	Software Evolution AND Code Smell Detection	2010-Sekarang	Ya	5
Science Direct				6
	Total			11
IEEExplore	Bad Smell Detection AND God Class	2010-Sekarang	Ya	0
Science Direct				16
	Total			16
IEEExplore			Ya	2

Science Direct	Code Smell Detection AND God Class	2010-Sekarang		12
	Total			14
IEEExplore	Software Quality AND God Class	2010-Sekarang	Ya	1
Science Direct				4
	Total			5
Total		62		



Gambar 1. Diagram hasil penyaringan literatur

Dari seleksi yang dilakukan berdasarkan tahun publikasi, tipe *paper*, dan kesesuaian judul maupun abstrak dengan topik didapatkan jumlah *paper* yang akan digunakan untuk dilakukan analisa sebanyak 10 buah *paper*. Penyeleksian *paper* dilakukan dengan tahun publikasi mulai 2010 sampai saat ini (2014). Kemudian dari jumlah yang didapat dari seleksi pertama dilakukan seleksi referensi, yang diambil adalah *paper* jurnal ilmiah dan *paper* konferensi. Hal ini dilakukan karena penulis memandang bahwa topik yang akan dibahas seharusnya berkaitan dengan perkembangan beberapa tahun terakhir, dan pastinya penelitian yang berkembang beberapa tahun terakhir sudah mencakup penelitian pada tahun-tahun sebelumnya.



Gambar 2. Skema pembagian kategori metode deteksi God Class

Tabel 5. Daftar literatur utama

Literatur	
Detecting Bad Smell in Source Code Using Change History Information	(Palomba, et al., 2013)
DECOR: A method for the specification and detection of code and design smells	(Moha, et al., 2010)
Design Defect Detection and Correction by Example	(Kessentini, et al., 2011)
Mining Version History for Detecting Code Smell	(Palomba, et al., 2014)

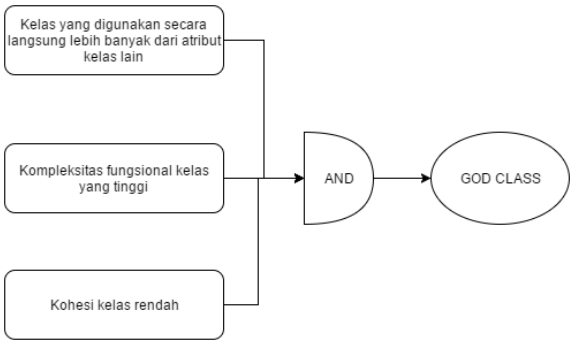
Setelah itu penyeleksian didasarkan pada kesesuaian literatur yang ada dengan topik yang dibahas, kemudian dilanjutkan dengan kesesuaian topik dengan abstraksi, dan isi secara umum. Literatur mengenai metode deteksi God Class yang akan digunakan dalam *review* ini adalah metode-metode yang tergolong *semi-automated* atau *automated*. Sedangkan metode yang tergolong manual tidak digunakan dalam *review*. Seleksi studi atau dokumen pada dasarnya terdiri dari beberapa tahap. Menghilangkan kesamaan isi atau metode antar dokumen dan melakukan penyaringan yang lebih dalam mengenai faktor-faktor eksklusi dan inklusi yang telah didefinisikan tidak dapat dilakukan hanya dengan memilah berdasarkan tahun publikasi dan jenis *paper*. Oleh karena itu, perlu dilakukan studi lebih dalam dengan mengkaji judul, abstrak, metode, hasil, dan konklusi dari setiap literatur sehingga didapatkan literatur yang paling sesuai untuk kemudian dapat di-*review*.

Beberapa dokumen pada Tabel 5 merupakan hasil dari studi kualitas dan literatur-literatur ini yang akan dijadikan literatur utama untuk melakukan *review*.

4. GOD CLASS

God Class dapat diartikan sebagai kelas yang menjadi pusat dari sistem. Kelas ini menjalankan sebagian besar pekerjaan dan hanya mendelegasikan peran-peran kecil pada kelas lain. Kelas ini juga banyak menggunakan data dari kelas lain.

Pendeteksian God Class memiliki 3 faktor, yaitu penggunaan kelas secara langsung yang lebih banyak daripada beberapa atribut dari kelas lain, kompleksitas fungsional sebuah kelas yang tinggi, dan kohesi antar kelas rendah. Dari ketiga faktor tersebut apabila dipenuhi, maka sebuah kelas dapat dikatakan terdapat Code Smell jenis God Class.



Gambar 3. Faktor yang menjadi parameter untuk mendeteksi God Class

5. METODE DETEKSI GOD CLASS

5.1. DETEX

DETEX adalah sebuah teknik deteksi yang memungkinkan *software engineers* untuk mendefinisikan Code Smell secara spesifik pada abstraksi tingkat tinggi menggunakan *unified vocabulary* dan *domain-specific language* yang didapatkan dari domain analisis. Teknik ini digunakan untuk menghasilkan algoritma deteksi secara otomatis. Berikut ini adalah langkah-langkah dalam melakukan deteksi:

- *Domain analysis*

Pada langkah ini dilakukan analisa secara mendalam terhadap domain dari Code Smell untuk menentukan konsep utama dari deskripsi yang diberikan. Dari konsep utama tersebut didapatkan *vocabulary*, *taxonomy*, dan klasifikasi dari Code Smell. Pada Gambar 4 dan Gambar 5 adalah contoh konsep utama dan *taxonomy*.

- *Specification*

Spesifikasi dilakukan menggunakan Domain Specification Language (DSL) dalam bentuk *rule card* berdasarkan *vocabulary* dan *taxonomy* yang didapatkan dari proses sebelumnya. *Rule card* mendefinisikan properti yang harus dimiliki oleh sebuah kelas untuk dapat digolongkan sebagai Code Smell. Penggunaan DSL memungkinkan software engineer atau expert untuk menentukan sendiri spesifikasi *rule* dari Code Smell.

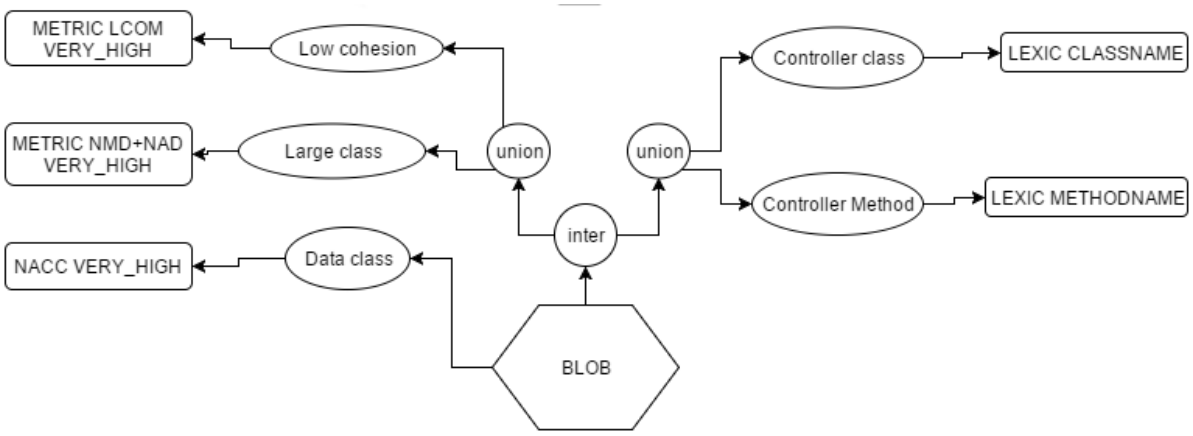
- *Algorithm Generation*

Algoritma dibuat secara otomatis menggunakan metamodel dan *parser*. Sebuah *framework* digunakan untuk memungkinkan algoritma dibuat secara otomatis.

- *Detection*

Deteksi dilakukan pada model dari sistem yang didapatkan baik melalui proses *forward engineering* ataupun *reverse engineering* pada sebuah sistem.

Validasi pada metode ini dilakukan oleh *independent engineer* untuk memastikan apakan sebuah kelas memang merupakan Code Smell. Subjek validasi adalah 4 *antipattern* beserta spesifikasinya yang didapatkan dari (Brown, et al., 1998) dan (Fowler, 1999). Spesifikasi yang didapatkan tersebut kemudian diproses secara otomatis untuk membentuk algoritma deteksi.Validasi dilakukan terhadap model hasil *reverse engineering* dari 10 *open source* sistem dengan bahasa pemrograman java, antara lain: ARGOUML, AZUREUS, GANTTPROJECT, LOG4J, LUCENE, NUTCH, PMD, QUICKUML, dan 2 versi dari XERCES. Hasil validasinya menunjukkan bahwa metode ini memiliki presisi sebesar 88.6% dan recall sebesar 100%.



Gambar 4. Contoh taksonomi metode DETEX

The Blob (also called God class) corresponds to a large controller class that depends on data stored in surroundung data classes. A large class declares many fields and methods with a low cohesion. A controller class monopolises most of the processing done by a system, takes most of the decisions, and closely directs the processing of other classes. Controller classes can be identified using suspicious names such as Process, Control, Manage, System, and so on. A data class contains only data and performs no processing on these data. It is composed of highly cohesive fields and accesors.

Gambar 5. Contoh konsep utama metode DETEX

```
RULE_CARD : Blob {
   RULE : Blob {ASSOC: associated FROM: mainClass ONE TO: DataClass MANY } ;

   RULE : mainClass {UNION LargeClassLowCohesion ControllerClass } ;

   RULE : LargeClassLowCohesion {UNION LargeClass LowCohesion } ;

   RULE : LargeClass { (METRIC: NMD + NAD, VERY_HIGH, 20) } ;

   RULE : LowCohesion { (METRIC: LCOM5, VERY_HIGH, 20) } ;

   RULE : ControllerClass {UNION (SEMANTIC: METHODNAME, {Process, Control, Ctrl,
Command, Cmd, Process, Proc, UI, Manage, Drive})
                                 (SEMANTIC: CLASSNAME,
{Process, Control, Ctrl, Command, Cmd, Process, Proc, UI, Manage, Drive, System,
Subsystem})} ;

   RULE : DataClass {(STRUCT: METHOD_ACCESSOR, 90) } ;

} ;
```

Gambar 6. Rule card yang digunakan untuk deteksi God Class

5.2. History Changes Analysis

Metode ini dijalankan berdasarkan *history data* pada setiap perubahan yang dilakukan dalam sebuah program. Hipotesa yang diajukan adalah bahwa ketika sebuah perubahan terjadi, kelas yang tergolong Blob atau God Class juga akan mengalami perubahan sebanyak α% dan melibatkan paling tidak satu kelas lain. HIST memanfaatkan analisa terhadap histori program untuk mendapatkan data-data yang diperlukan dalam melakukan deteksi. Untuk melakukan deteksi terhadap Blob atau God Class digunakan *rule card* pada Gambar 6.

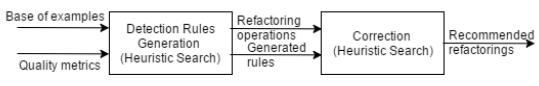
5.3. Detection Defect by Example

Metode ini memanfaatkan *repository* yang berisi data *code* yang sudah digolongkan sebagai defect atau smell. Dari data yang didapat kemudian dihasilkan aturan pendeteksian berbentuk kombinasi dari matriks kualitas perangkat lunak. Skema pendeteksiannya dapat dilihat pada Gambar 8. Metode ini memanfaatkan *Genetic Programming* untuk mencari aturan yang sesuai. Populasi pada algoritma ini merepresentasikan kemungkinan aturan yaitu berupa kombinasi matriks kualitas. Seperti algoritma genetic pada umumnya, populasi pertama yang telah terbentuk dihitung nilai *fitness*-nya. Setelah itu, individu dengan nilai *fitness* tertinggi akan digunakan dalam proses *crossover*. Hasil dari proses *crossover* kemudian dimutasi untuk menghasilkan populasi yang baru. Proses ini dilakukan sampai memenuhi batas iterasi dan dihasilkan aturan terbaik untuk melakukan deteksi. Fungsi yang digunakan untuk melakukan perhitungan *fitness value* pada metode ini dapat dilihat pada persamaan dibawah ini.

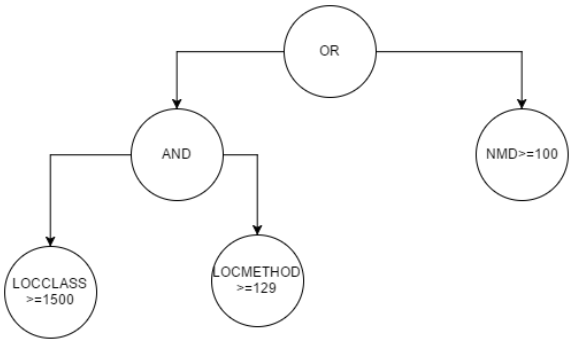
Pada Persamaan (1) t adalah jumlah kelas yang dideteksi, p adalah jumlah kelas contoh yang tergolong *bad smell*. a bernilai 1 ketika kelas yang terdeteksi terdapat pada contoh, dan sebaliknya untuk a yang bernilai 0.

Validasi untuk metode ini dilakukan terhadap 4 *open source* sistem dengan bahasa pemrograman java antara lain, Xerces-J, ArgoUML,Quick UML, dan GanttProject. Prosedur validasinya menggunakan 4-*fold cross validation*, yaitu teknik model validasi untuk menilai hasil analisis statistikal yang menghasilkan dataset independen. Hasil deteksi sebuah sistem dievaluasi menggunakan 3 sistem lain sebagai contoh *defect*. Nilai presisi dan *recall* yang dihasilkan dari metode ini sebesar 96% dan 99.5%.Pada penelitian ini validasi dilakukan oleh beberapa mahasiswa pascasarjana untuk memastikan hasil deteksi dengan dasar definisi dari setiap Code Smell. Deteksi dilakukan terhadap *git history* dari 8 *project* java antara lain: Apache Ants, Apache Tomcat, Jedits, dan 5 *project* Android API. Hasil deteksi menunjukkan tingkat presisi dan *recall* dari metode sebesar 76% dan 61%.

$$f_{norm} = \frac{\frac{\sum_{i=1}^{p} a_i}{t} + \frac{\sum_{i=1}^{p} a_i}{p}}{2} \quad (1)$$



Gambar 7. Skema pendeteksian *defect*



Gambar 8. Tree yang digunakan untuk merepresentasikan *detection rule*

6. Hasil Komparasi

Dapat dilihat pada Tabel 6 bahwa metode dengan nilai presisi dan *recall* terbaik berhasil didapatkan oleh metode ketiga, yaitu pendeteksian God Class secara otomatis berdasarkan contoh kelas yang terdeteksi manual sebagai *defect*. Berdasarkan studi yang dilakukan, hal ini disebabkan pembangunan aturan pendeteksian dilakukan berdasarkan contoh kelas yang sudah dideteksi sebagai God Class. Semakin baik dan lengkap data yang digunakan sebagai contoh maka aturan pendeteksian akan menjadi semakin lengkap dan mampu mendeteksi dengan lebih baik. Namun hal ini juga yang menjadi kekurangan utama dari metode ini, yaitu aturan yang dihasilkan bergantung pada data training yang digunakan.

7. DAMPAK DARI GOD CLASS

God Class berpotensi membahayakan desain sebuah sistem karena berupa agregasi dari berbagai abstraksi dan kelas-kelas lain baik yang digunakan maupun tidak. Hal ini seringkali dikarenakan banyaknya penggunaan kelas yang tidak mematuhi kaidah pemrograman berbasis objek, dimana satu kelas seharusnya hanya memiliki sebuah tugas tertentu.

Dari implementasi yang tidak mengikuti kaidah pemrograman berbasis objek akan mengakibatkan terhambatnya evolusi perangkat lunak karena struktur perangkat lunak akan sulit dimengerti dan dirawat kedepannya. Dampak lainnya adalah tingginya biaya perawatan perangkat lunak, hal ini diakibatkan sulitnya perawatan yang berkorelasi dengan tingginya biaya perawatan.

8. KESIMPULAN

Studi literatur dilakukan untuk mengetahui metode-metode pendeteksi God Class dan dampak dari God Class tersebut pada sistem. Dari proses seleksi literatur yang dilakukan, didapatkan 3 metode untuk mendeteksi salah satu jenis Bad Smell, God Class. Metode-metode tersebut adalah metode pendeteksian dengan dasar pendeteksian berupa aturan deteksi. Perbedaan antar metode tersebut adalah proses pendeteksiannya, terdapat metode yang dapat mendeteksi God Class secara semi-otomatis dan terdapat juga yang dapat mendeteksi

Tabel 6. Hasil komparasi metode

Literatur	Metode	Hasil	Input & Output	Dataset	Imple-mentasi	Kekurangan	Kelebihan
(Moha, et al., 2010) (Palomba, et al., 2013) (Olbrich & Cruzes, 2010)	Algoritma deteksinya dibuat dalam bentuk *rule card* menggunakan DSL yang didapatkan dari analisis domain setiap bad smell secara mendalam.	*Precision*: 52% *Recall*: 49%	Input: Deskripsi mengenai Code Smell Output:	10 Open source java system	Ya	Pembangunan *rule card* secara manual membutuhkan usaha yang lebih	*Expert* dan *engineer* dapat dengan mudah menspesifikasi-kan aturan pendeteksian.
(Palomba, et al., 2013) (Olbrich & Cruzes, 2010) (Palomba, et al., 2014)	Memanfaatkan *rule card* pada (Moha, et al., 2010) dan data perubahan yang terjadi pada software.	*Precision*: 76% *Recall*: 61%	Input: Versioning system address Output: Affected Components	8 Java Project	Ya	Tidak bisa mendeteksi kelas yang belum pernah mengalami perubahan	Nilai presisi yang lebih baik dari algoritma sebelumnya.
(Kessenti ni, et al., 2011)	Membangun aturan pendeteksian secara otomatis berdasarkan contoh kelas yang terdeteksi manual sebagai *defect*.	*Precision*: 96% *Recall*: 99.5%	Input: Contoh *defect class* & quality metrics Output: Generated rules	4 Open source java project	Tidak	Aturan-aturan yang dihasilkan sangat bergantung pada data training	Aturan pendeteksian dibangun secara otomatis.

secara otomatis. Pada metode DECOR (Moha, et al., 2010) menggunakan cara deteksi semi-otomatis dalam pembentukan algoritma deteksi dari rule card. Metode ini menghasilkan prosentase presisi sebesar 88.6% dan recall 100%. Metode kedua (Palomba, et al., 2013) menggunakan rule card dan catatan histori perubahan pada sebuah perangkat lunak. Hasil yang didapat adalah prosentase presisi sebesar 76% dan recall 61%. Metode ketiga (Kessentini, et al., 2011) adalah pendeteksian berdasarkan contoh class yang dideteksi manual sebagai defect perangkat lunak. Hasil dari metode ini adalah yang terbaik dibandingkan dengan kedua metode sebelumnya yaitu dengan prosentase presisi 96% dan recall 99.5%. God Class memiliki dampak terhadap *understandability* dan *maintainability* yang mempengaruhi biaya yang dikeluarkan untuk perawatan.

9. DAFTAR PUSTAKA

BROWN, W. J., MALVEAU, R. C. & MOWBRAY, T. J., 1998. Anti Patterns: Refactoring Software, Architectures, and Project in Crisis. s.l., s.n.

DE F. CARNEIRO, G. et al., 2010. Identifying Code Smells with Multiple Concern Views. s.l., s.n.

FONTANA, F. A. et al., 2013. Investigating the Impact of Code Smells on System's Quality: An Empirical Study on Systems of Different Application Domains. s.l., s.n.

FOWLER, M., 1999. Refactoring-Improving the Design of Existing Code. s.l., s.n.

HAMID, A. & ILYAS, M., 2013. A Comparative Study on Code Smell Detection Tools. s.l., s.n.

KESSENTINI, M., KESSENTINI, W. & SAHRAOUI, H., 2011. Design Defect Detection and Correction by Example. s.l., s.n.

LANZA, M., MARINESCU, R. & S, D., 2005. Object Oriented Metrics in Practice. New York: Springer.

MOHA, N., GUE´HE´NEUC, Y.-G. & DUCHIEN, L., 2010. DECOR: A method for the specification and detection of code and design smells. s.l., s.n.

OLBRICH, S. M. & CRUZES, D. S., 2010. Are all Code Smells Harmful? A Study of God Classes and Brain Classes in the Evolution of three Open sources Systems. s.l., s.n.

PALOMBA, F., BAVOTA, G. & PENTA, M. D., 2013. Detecting Bad Smell in Source Code Using Change History Information. s.l., s.n.

PALOMBA, F., BAVOTA, G. & PENTA, M. D., 2014. Mining Version Histories for Detecting Code Smells. s.l., s.n.

SJØBERG, D. I. et al., 2013. Quantifying the Effect of Code Smells on Maintenance Effort. IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, Volume 39, pp. 1144-1196.

ZHANG, M. & HALL, T., 2011. Code Bad Smells: a review of current knowledge. Journal of Software Maintenance and Evolution, 23(3), pp. 179-202 .

FILKOM
FAKULTAS ILMU KOMPUTER

JURNAL TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA